PeopleSoft

EnterpriseOne 8.9 インタオペラビリティ PeopleBook

2003 年 9 月

PeopleSoft EnterpriseOne 8.9
インタオペラビリティ PeopleBook
SKU AC89JIO0309

オープンソースの開示

目次

略称

ERP で一般に使用され、『インタオペラビリティ』ガイドに記されている略称（頭文字）のリストを次に示します。

APAg	拡張プランニング・エージェント
API	アプリケーション・プログラム・インターフェイス
APPL	アプリケーション
BDA	ビジネス・ビュー設計ツール
BSFN	ビジネス関数
BSVW	ビジネス・ビュー
COM	コンポーネント・オブジェクト・モデル
CRP	カンファレンス・ルーム・パイロット
DBMS	データベース管理システム
DCOM	分散コンポーネント・オブジェクト・モデル
DD	データ辞書
DLL	ダイナミック・リンク・ライブラリ
DS または DSTR	データ構造体
EDI	電子データ交換
ER	イベント・ルール
ERP	ERP（基幹業務統合パッケージ）
FDA	フォーム設計ツール
IDL	インターフェイス定義言語
NER	イベント・ルール・ビジネス関数
ODA	Open Data Access
ODBC	オープン・データベース・コネクティビティ
OCM	オブジェクト構成マネージャ(OCM)

OL	オブジェクト・ライブラリアン
OSA	OSA（Output Stream Access/出力ストリーム・アクセス）
QBE	QBE
RDA	レポート設計ツール
SAR	ソフトウェア・アクション・リクエスト
Specs	スペック
SQL	構造化照会言語
TAM	テーブル・アクセス管理
TBLE	テーブル
TC	テーブル変換
TDA	テーブル設計ツール
TER	テーブル・イベント・ルール
UBE	ユニバーサル・バッチ・エンジン
WF	ワークフロー
XAPI	拡張アプリケーション・プログラム・インターフェイス
XML	拡張可能マークアップ言語
XPI	XPI(eXtended Process Integration)
XTS	XML 変換サービス

インタオペラビリティの概要

インタオペラビリティは、通常、散在するアプリケーションをまとめ挙げることのできる手段としてソフトウェアに関連付けられています。たとえば、インタオペラビリティにより、企業はさまざまなベンダのアプリケーションを単一ベンダのアプリケーションであるかのように使用できます。機能と情報のシームレスな共有が可能になります。

また、オートメーションに伴う問題が減少したり、解消されます。アプリケーション間でのビジネスのプロセス・フローが実現します。さらに、インタオペラビリティによりシステムを相互に連動させ、重要なビジネス情報をリアルタイムで共有できます。インタオペラビリティ・オプションは、システムとアプリケーションを結ぶ役割を果たします。

ERP インタオペラビリティ機能

システム間のデータ・フローは、完全なインタオペラビリティによって、ユーザーから見てシームレスとなります。J.D. Edwards では、外部システムとのインタオペラビリティの複雑性を隠し、サードパーティのパッケージとのインターフェイスを簡素化するための基盤構造を提供しています。

ERP インタオペラビリティ・ソリューションは、次の 3 つの重要なビジネス目的に適合しています。

- 柔軟かつ豊富な選択肢

  J.D. Edwards では、ERP で処理するアプリケーションおよびデータのタイプをいくつか用意しています。既存ツール、最良品質ツール、顧客管理ツール、レポート作成ツールなどがあります。開発者は、特定環境およびニーズに合わせて正しい選択を行うことができます。

- 投資保護

  引き続き使用したい既存のアプリケーションがあるか、または採用を考えているアプリケーションがある場合、そのアプリケーションと ERP とのインターフェイスをとることができます。既存のテクノロジーまたは新しいテクノロジーがアプリケーションをサポートしている場合は、業界標準方法を使用できます。または、J.D. Edwards ビジネス・ロジックを使用すると、このインタオペラビリティをもたらすことができます。そのアーキテクチャに対するアップグレードや改善も可能です。

- 管理可能性

  ERP は、インタオペラビリティ・プロセスを簡単に管理できるように設計されています。

利点

インタオペラビリティには、次の利点があります。

- 製品開発元や種類に関係なく、企業全体のアプリケーションおよびシステムをまとめることができます。
- ビジネス実行コストを減らすか、または競争力を高めるために、取引先同士の協力関係を生み出すことができます。

- 複数のシステムを共にリンクさせ、リアルタイムでデータを共有し、時間に左右されやすいデータを必要とするユーザーに即座に提供できます。
- 合併や買収によって生じた異なるソリューションを迅速に統合し、企業の情報テクノロジー・ソリューションにまとめ上げることができます。

J.D.・Edwards インタオペラビリティ戦略には、広範なモデルと機能（業界標準および ERP ベース）があります。

インタオペラビリティ・モデルおよび機能

次の表は、J.D. Edwards がサポートしているインタオペラビリティ・モデルの概要を示します。この表に示すモデルはさらに数タイプに分かれており、モデル・タイプごとに使用できる機能です。モデルとモデル・タイプは、左端の欄に示してあります。機能とは ERP データへのアクセス方法で、表の各列として示されています。モデル・タイプごとに、どのような機能を使用できるかを右の欄で確認できます。J.D. Edwards では、対話型機能とバッチ機能の両方を用意しています。機能はインバウンド、アウトバウンド、およびバッチにグループ化されています。インバウンド機能は、ERP 外部で開始されたデータ要求またはトランザクション要求です。アウトバウンド機能は、ERP 内部で実行されます。

ERP インタオペラビリティ									
	ERP への受信（インバウンド）				ERP からの送信（アウトバウンド）				バッチ
機能 / モデル	BSFN の呼出し	XML CallObj.、XML List、XML Trans.	Z Trans.	フラット・ファイル	イベント: リアルタイム・イベントと XAPI イベント	イベント: Z イベント	XML 出力の生成	フラット・ファイル	バッチの推奨
XPI									
エンタープライズ XPI	Y	リスト	N	Y	Y	Y	Y	Y	N
エンタープライズ間 XPI	Y	リスト	N	Y	Y	Y	Y	Y	N
コネクタ									
動的 Java コネクタ（Java コネクタ）	Y	CO、Trans、List	N	N	Y	Y	Y	N	N
JCA リソース・アダプタ	Y	CO、Trans、List	N	N	N	N	N	N	N
COM コネクタ	Y	CO、Trans、List	N	N	Y	Y	Y	N	N
ERP メッセージング・アダプタ									
MQSeries 用アダプタ	Y	CO、Trans	Y	N	Y	Y	Y	N	Y
MSMQ 用アダプタ	Y	CO, Trans	Y	N	Y	Y	Y	N	Y

バッチ・インターフェイス									
インターフェイス・テーブル	Y	N	Y	Y	N	N	N	N	Y
ERP EDI	Y	N	Y	Y	N	N	N	Y	Y
テーブル変換	Y	N	Y	Y	N	N	N	Y	Y
OSA (UBE)	N	N	N	N	N	N	Y	N	Y
APAg/インテグレーション	N	N	Y	N	Y	N	Y	N	Y
Open Data Access	ビジネス・ビューとテーブルの照会のサポート				N				N

インタオペラビリティ機能

機能とは、ERP との間で情報を転送したり取得したりする手段です。インタオペラビリティの表は、受信機能と送信機能、およびバッチ処理に該当する機能を示します。受信機能を使用すると、データの照会と更新（追加、変更、または削除）が可能です。照会機能は、参照のためにのみデータを取得する場合に使用します。たとえば、品目の価格や在庫状況を知りたい場合があります。更新機能は、トランザクションごとに個別に実行するか、トランザクション・グループで構成されるバッチ処理として実行できます。個々のトランザクションの更新では、購買オーダーの追加や請求書の作成など、1 つのレコードが更新されます。バッチ処理とはトランザクションのグループであり、通常は複数のレコードが更新されます。このバッチ処理は非対話型で、一般に特定の時刻に実行されるようにスケジュールされます。バッチ処理の例には、1 日の終わりにデータベースに 10,000 件のオーダーをアップロードする処理や、変更があった価格設定情報を 1 日の終わりにすべて取得して Web サイトに送信する処理などがあります。

ここでは、ERP との間で情報を転送および取得するために使用可能な機能の概要について説明します。個々の機能については、他の箇所で詳しく説明します。

ビジネス関数の呼出し

ビジネス関数の呼出しは、J.D. Edwards インタオペラビリティの核心部です。ビジネス関数により、仕訳トランザクション、減価償却計算、および受注オーダー・トランザクションなど、特定のタスクを実行するためのトランザクション・ロジックがカプセル化されます。

J.D. Edwards のビジネス関数には、通常のビジネス関数およびマスター・ビジネス関数(MBF)の 2 種類があります。通常のビジネス関数は、税計算または勘定科目コード確認などの簡単なタスクを実行します。マスター・ビジネス関数(MBF)は、より複雑なタスクを実行するもので、そのタスクを実行するためにいくつかのビジネス関数を呼び出すことができます。

参照

- ビジネス関数については『開発ツール』ガイドの「ビジネス関数」

XML

XML は、システム間で情報を共有し、データを移動するための、柔軟性を備えた標準ベースの手段です。XML により、企業のアプリケーションの拡張や、ビジネス・パートナーおよび顧客との共同作業が可能になります。XML CallObject および XML Transaction を使用して、ERP データを更新また

は取得できます。XML List を使用すると、ERP システム・リポジトリに XML データ・ファイルを作成し、ネットワーク・トラフィックを回避するために小さい単位でデータを取得できます。ERP の出力は XML ドキュメント形式です。

Z トランザクション

Z トランザクションは ERP に受信機能を提供し、ERP データの更新を可能にします。J.D.Edwards は、Z トランザクション機能をサポートするインターフェイス・テーブル（Z テーブル）を提供しています。独自の Z テーブルを作成することもできます。

フラット・ファイル

フラット・ファイルは、通常はワークステーションまたはサーバーに保管されるテキスト・ファイルで、ユーザー定義フォーマットと呼ばれることもあります。フラット・ファイルにはリレーションシップは定義されておらず、通常は Unicode キャラクタ・セットが使用されます。フラット・ファイル内のデータは、連続した情報の文字列として保管されています。フラット・ファイルを使用すると、対話方法が他にないアプリケーションからデータをインポートまたはエクスポートできます。たとえば、ERP と他のアプリケーション間でデータを共有したい場合があります。

イベント

イベントとは、サードパーティ・アプリケーションまたはエンドユーザーに対して ERP ビジネス・トランザクションが発生したことを知らせる通知です。J.D. Edwards は、Z イベント、リアルタイム・イベント、XAPI イベントの 3 種類をサポートしています。イベント・データは XML ドキュメントとして表されます。

Z イベントでは、インターフェイス・テーブルとバッチ処理を使用してトランザクション情報が取得され、Z イベント・ジェネレータとデータ・エクスポート・サブシステムを使用して送信データ・フローが管理されます。

リアルタイム・イベントは、サーバーまたはクライアントで生成できます。システム呼出し（サーバーから）とクライアントのビジネス関数の呼出し（クライアントから）により、トランザクション情報が取得されます。このトランザクション情報は、サブスクライバに配布されます。

XAPI イベントは、応答を必要とするリアルタイム・イベントです。この種のイベントはリアルタイム・イベントと同じ方法で作成されますが、応答 XML Document の受信時にビジネス関数を呼び出すためのデータ構造体情報が追加されます。

XPI(eXtended Process Integration)

XPI は、Extended Process Integration スイートです。XPI は、エンタープライズ XPI とインターエンタープライズ XPI という 2 つの主要製品で構成されます。

- Enterprise XPI では、企業内の内部コラボレーション用のインフラストラクチャが提供されます。
- インターエンタープライズでは、企業間のビジネス間(B2B)コラボレーション用のインフラストラクチャが提供されます。

エンタープライズ XPI を使用する利点は、次のとおりです。

- 多数のアダプタが存在します。
- セキュリティと許可のために共通のアクセス・ポイントが提供されます。

- ブローカとアダプタの追加により、スケーラビリティが実現します。
- メッセージ配信が保証されます。
- インテグレーションとワークフローを作成するためのグラフィック・ツールが存在します。
- 定義済みの拡張ビジネス・プロセス(XBP)と標準により、J.D. Edwards データとのインテグレーションが容易になります。

インターエンタープライズ XPI を使用する利点は、次のとおりです。

- ベンダや取引先などの外部ソースにインタオペラビリティを提供します。
- 保証付きの配信およびセキュリティを提供します。
- 標準的な XML、EDI、OAG、RosettaNet、BizTalk、eXML、ebXML など、業界標準フォーマットを使用します。
- ワークフローおよびマッピング機能が用意されています。
- RosettaNet 用にパッケージされたトランザクションが使用可能です。

XPI の使用法については、J.D. Edwards Knowledge Garden でエンタープライズ XPI およびインターエンタープライズ XPI のドキュメンテーションを参照してください。使用可能なドキュメンテーションには、エンタープライズ XPI の基礎、インターエンタープライズ XPI の基礎、アダプタおよび XBP に関する情報が記載されています。

インタオペラビリティ・モデル

モデルは、サードパーティを ERP システムに接続またはアクセスさせるための手段です。
J.D Edwards は、次の 5 つの基本的なインタオペラビリティ・モデルをサポートしています。

- XPI (eXtended Process Integration)
- コネクタ
- メッセージング・アダプタ
- バッチ・インターフェイス
- Open Data Access
- この 5 つのモデルは、さらにタイプ別に分類されます。各モデル・タイプは、ERP データベースとの情報の送信や取得について複数の J.D. Edwards 機能をサポートしています。「インタオペラビリティ・モデルおよび機能」で紹介した表は、モデル・タイプと各モデル・タイプでサポートする機能を示します。ここでは、各モデル・タイプの概要について説明します。

コネクタ

コネクタは 2 点間コンポーネント・ベースのモデルであり、サードパーティ・アプリケーションと ERP でロジックとデータの共有を可能にします。J.D. Edwards コネクタ・アーキテクチャには、Java コネクタと COM コネクタが組み込まれています。コネクタは、受信 XML 要求を受け入れて、ERP ビジネス関数を再利用できるように公開します。コネクタからの出力は、XML ドキュメント形式です。

- Java – J.D. Edwards の動的 Java および Java コネクタは、リアルタイム・イベント処理をサポートしています。Java は移植可能な言語であるため、ERP 機能を Java アプリケーションに容易に連結できます。

- COM – J.D. Edwards の COM コネクタ・ソリューションは、Microsoft のコンポーネント・オブジェクト・モデルに全面的に準拠しています。ERP 機能を Visual Basic および VC++アプリケーションに容易に連結できます。また、COM コネクタは、リアルタイム・イベント処理をサポートしています。

コネクタを使用する利点は、次のとおりです。

- スケーラビリティを備えています。
- マルチスレッド化するように設計されています。
- 同時ユーザーが可能です。

参照

J.D. Edwards コネクタについては『コネクタ』ガイドの「コネクタ」およびその他の項

メッセージング・アダプタ

J.D. Edwards は、MQSeries および Microsoft Message Queuing(MSMQ)のメッセージ処理サポートを提供しています。MQSeries と MSMQ は、メッセージの待ち行列処理、メッセージ配信、およびトランザクションのモニタリングを取り扱います。ERP では、これらのメッセージ処理システムを使用して、ERP とサードパーティ・アプリケーションの間でロジックおよびデータ要求を処理し、受け渡すことができます。

メッセージング・アダプタを使用する利点は、次のとおりです。

- 信頼性の高い接続
- 保証付き配信
- 操作の確認

参照

- メッセージング・アダプタの使用方法については、『Asynchronous Messaging Programmer's Guide(非同期メッセージ処理プログラマ・ガイド)』の「Adapter Overview(アダプタの概要)」およびその他の項

バッチ・インターフェイス

バッチとは、同時に複数のトランザクションが処理されることを意味し、通常は大量の情報移動を伴います。バッチ処理は通常、スケジュールされており、非対話型です。J.D. Edwards では、バッチ処理用に複数のモデル・タイプを用意しており、各モデル・タイプには ERP データへのアクセスに使用できる 1 つまたは複数の機能があります。次のモデル・タイプがあります。

- インターフェイス・テーブル
- 電子データ交換(EDI)
- テーブル変換
- UBE
- APAg/インテグレーション

インターフェイス・テーブル

インターフェイス・テーブルは、データのインポートとエクスポートのための 2 点間インタオペラビリティ・ソリューションを提供します。このテーブルは Z テーブルと呼ばれることもあります。インターフェイス・テーブルは作業ファイルであり、ここにトランザクション情報を入れて ERP との間で処理します。J.D. Edwards が提供するインターフェイス・テーブルの他に、独自のインターフェイス・テーブルを作成することもできます。インターフェイス・テーブルを使用して ERP データを更新する場合、そのデータには J.D. Edwards の定義済みフォーマットを使用する必要があります。インターフェイス・テーブルを使用して ERP データを取得する場合は、アプリケーション・テーブルからデータを抽出するバッチ処理を使用します。

インターフェイス・テーブルを使用する利点は、次のとおりです。

- 定義済みのデータ構造体
- 識別可能なフィールド
- カスタマイズ可能なインターフェイス・テーブル

電子データ交換（EDI）

EDI は、データのインポートとエクスポートのための 2 点間インタオペラビリティ・ソリューションを提供します。EDI とは、標準的な内容を持つ標準フォーマットでの購買オーダーや請求書など、ビジネス・トランザクションをペーパーレスに、コンピュータ間でやりとりすることです。このように電子データ交換は、電子商取引戦略の中核といえます。

EDI を使用してデータをやりとりする場合、同じ EDI 標準フォーマットを使用する相手先のシステムが認識できるように、データは EDI 標準フォーマットで伝送されます。送られてきた EDI 標準フォーマットのデータは社内システムのフォーマットに変換しなければならないため、EDI を使用する会社は、独自の変換ソフトウェアを備えている必要があります。

電子データ交換システム用 J.D. Edwards データ・インターフェイスは、J.D. Edwards システム・データと変換ソフトウェア間のインターフェイスとして機能します。このデータ・インターフェイスは、EDI データ交換の他に、一般のインタオペラビリティにも使われます。電子商取引では、ファイル基準のインターフェイスがビジネス要件を満たしている必要があります。

電子データ交換システム用のデータ・インターフェイスを使用する利点は次のとおりです。

- 履行サイクルを短縮する。
- 手作業データ入力を減らすことによりデータ整合性を増す。
- 事務の手作業を減らす。

EDI は、複数のアプリケーションに同時に情報を送信する場合に特に効果的です。

参照

- インタオペラビリティに電子データ交換(EDI)を使用する方法については、『Data Interface for Electronic Data Guide（電子データ用データ・インターフェイス）』ガイドの「Overview for Data Interface for Electronic Data Interchange System（電子データ交換システム用のデータ・インターフェイスの概要）」およびその他の項

テーブル変換

テーブル変換は、データのインポートとエクスポートのための 2 点間インタオペラビリティ・ソリューションを提供します。テーブル変換とは、テーブル内のデータを高速処理できる特殊な形式のユニバーサル・バッチ・エンジン(UBE)です。ERP には、企業データの収集、フォーマット、インポートおよびエクスポートに使用できるテーブル変換ユーティリティがあります。テーブル変換ツールで、データの転送やコピーができます。また、テーブルからレコードを削除することもできます。テーブル変換により、J.D. Edwards 以外のテーブルを使用してビジネス関数を処理し、直接呼び出して出力できます。たとえば、ERP マスター・テーブルから読み取って ERP 以外のテーブルに自動入力する UBE を実行できます。

テーブル変換ユーティリティは、ERP テーブル、ビジネス・ビュー、テキスト・ファイル、または ERP 以外のテーブルであっても Oracle、Access、AS/400、または SQL Server などの ERP がサポートしているデータベースに常駐するテーブルに対して利用できます。これらの ERP 以外のテーブルは、通常、外部形式テーブルといいます。

参照

- テーブル変換については『テーブル・コンバージョン』ガイドの「テーブル変換の概要」およびその他の項

出力ストリーム・アクセス(OSA)

OSA は、UBE からデータをエクスポートするための 2 点間インタオペラビリティ・ソリューションを提供します。OSA を使用すると、ERP から Microsoft® Excel などの他のソフトウェア・パッケージにデータを渡して処理させるためのインターフェイスをセットアップできます。

OSA を使用する利点は、次のとおりです。

- 出力を手作業でフォーマットする必要がなくなります。
- ターゲット・ソフトウェア・プログラムの処理能力を利用できます。

参照

- 出力ストリーム・アクセス(OSA)の使用方法については、『エンタープライズ・レポート・ライティング』ガイドの「出力ストリーム・アクセス」

拡張プランニング・エージェント(APAg)/インテグレーション

J.D.Edwards の拡張プランニング・エージェント(APAg)は、企業データのバッチ抽出、変換、ロードを行うツールです。APAg は、リレーショナル・データベース、フラット・ファイル・フォーマットおよび XML のような他のデータまたはメッセージ・エンコーディング形式によるデータ・ソースへのアクセスをサポートしています。また、APAg はデータを 2 点間で移動し、そのデータ移動に関連するタスクを実行します。

APAg を使用する利点は、次のとおりです。

- 大量のテーブル・データをコピーできます。
- 初期データ・ロードを有効かつ効率的に行います。

参照

- APAg ツールについては、『Advanced Planning Agent 3_3_3 User Guide（拡張プランニング・エージェント 3_3_3 ユーザー・ガイド）』の「APAg Integrates Your System（APAg によるシステムの統合）」およびその他の項

オープン・データ・アクセス(ODA)

ODA には、SQL ステートメントを使用して ERP データを抽出する機能が用意されているため、情報を集計してレポートを生成できます。ODA は、次のすべてのデスクトップ・アプリケーションと共に使用できます。

- Microsoft Query
- Microsoft Access
- Microsoft Excel
- ODBCTEST
- Crystal Report
- Microsoft Analysis Service

ODA はフロントエンド・クエリー/レポート作成アプリケーションと ERP が構成する ODBC ドライバの間に置かれます。

J.D. Edwards ERP データベースには、データが正しく表示されるように変換または適用する必要があるオブジェクトおよびカラム名、特定のデータ・タイプおよびセキュリティ・ルールがあります。特定のデータ・タイプおよびルールには、小数点シフト、ユリウス暦日付、通貨、メディア・オブジェクト、セキュリティ、およびユーザー定義コードなどがあります。一部のインスタンスでは、選択したアプリケーション内でデータが正しく表示されるように、ODA がデータと同様に SQL の SELECT ステートメントを修正することもあります。

ODA を使用する利点は、次のとおりです。

- データ辞書全体を含め、すべての ERP データへの読取り専用アクセス
- ERP 用に設定したのと同じセキュリティ・ルールの使用
- ERP データを容易に抽出する機能

参照

- ODA の使用方法については、『インタオペラビリティ』ガイドの「オープン・データ・アクセス（ODA）」

インタオペラビリティ・モデルの選択

どのインタオペラビリティ・モデルを選択するかは、ビジネス・ニーズによって異なります。次の表を参考にして、貴社のインタオペラビリティ要件に最も適したインタオペラビリティ・モデルを判断してください。

ERP インタオペラビリティ					
考慮事項 / モデル	プラットフォーム (Windows、UNIX、AS/400)	インテグレーション・モデル	最適のプログラミング言語	受信トランザクションの作成に不可欠な技術上のスキル	送信トランザクションの作成に不可欠な技術上のスキル
XPI					
エンタープライズ XPI	SUN、AIX、Windows	ブローカ	Java	XPI ツールセット	XPI ツールセット、リアルタイム・イベント、XAPI イベント
インターエンタープライズ XPI	SUN、AIX、Windows	ブローカ	Java	XPI ツールセット	XPI ツールセット、リアルタイム・イベント、XAPI イベント
コネクタ					
Java コネクタ	すべて	2 点間	Java	JavaAPI、GenJava	リアルタイム・イベント、XAPI イベント
COM コネクタ	Windows	2 点間	C/C++/VB	COM、GenCOM	該当なし
メッセージング・アダプタ					
MQSeries 用アダプタ	すべて	ブローカ	HTML、C/C++、Java	MQSeries、XML	Z テーブル、サブシステム処理（R00460、データ・エクスポート制御など）
MSMQ 用アダプタ	Windows	ブローカ	C/C++	MSMQ、XML	Z テーブル、サブシステム処理（R00460、データ・エクスポート制御など）
バッチ・インターフェイス					
インターフェイス・テーブル	すべて	2 点間	任意	Z テーブル、UBE	カスタム・コード
ERP EDI	すべて	2 点間	任意、フラット・ファイル	Z テーブル、UBE	カスタム・コード
テーブル変換	すべて	2 点間	ERP TC	テーブル変換	テーブル変換ディレクタ/RDA
OSA (UBE)	すべて	2 点間	HTML、C/C++	該当なし	RDA、カスタム・コード
APAg/インテグレーション	UNIX、Windows	2 点間	独自言語	APAg ツール、Z テーブル	APAg ツール、Z テーブル
Open Data Access	すべて	2 点間	VB	カスタム・コードまたはサードパーティ・アプリケーション（クエリーのみ）	

その他の業界標準のサポート

ERP には、次のように、その他の業界標準機能をサポートするメディア・オブジェクト機能が用意されています。

- OLE（Object Linking and Embedding）：異なるデータ・タイプのデータ交換をする
- DDE（動的データ交換）：アプリケーション間の静的リンクと動的リンクを行う
- BLOB（バイナリ・ラージ・オブジェクト）タイプ：アプリケーション内でメディア・オブジェクトを添付する。
- MAPI（拡張メッセージング API）：異なるメールおよびグループウェア・アプリケーション間でのメッセージ交換を行う

参照

- 『基本操作』ガイドの「メディア・オブジェクト添付」
- 『システム・アドミニストレーション』ガイドの「メディア・オブジェクトとイメージング」
- 『基本操作』ガイドの「メッセージと待ち行列」

ビジネス関数の呼出し

ビジネス関数とは、複数のアプリケーションで再利用できる一連のビジネス・ルールとロジックをカプセル化したものです。ビジネス関数は、ERP データベースへの共通のアクセス手段となります。また、ビジネス関数により、特定のタスクが実行されます。マスター・ビジネス関数(MBF)は、トランザクション全体を拡張、編集してデータベースにコミットするために必要なロジックとデータベースの呼出しを提供します。サードパーティ・アプリケーションでは、ERP 機能全体、データ検証、セキュリティおよびデータ整合性の維持のためにマスター・ビジネス関数を使用できます。

マスター・ビジネス関数を使用すると、マスター・テーブル(住所録マスターや品目マスターなど)を更新したり、トランザクション・テーブル(受注オーダーや購買オーダーなど)を更新できます。通常、マスター・テーブルのマスター・ビジネス関数はテーブルにアクセスするもので、プログラム固有であるトランザクション・ファイルのマスター・ビジネス関数よりも単純です。トランザクション用マスター・ビジネス関数は、トランザクション・ファイルに必要なすべてのデフォルト値と編集機能を提供する共通の関数セットです。また、この種の関数はデータベースへの挿入、更新または削除されるトランザクションの整合性を維持するロジックを含みます。

インタオペラビリティを確保する場合は、テーブル入出力の代わりにマスター・テーブルのマスター・ビジネス関数を使用することで、テーブル・イベント・ルールの代わりにマスター・ビジネス関数を使用して関連テーブルを更新できます。複数のレコードは使用されず、代わりにすべての編集と操作が 1 回の呼出しで実行されます。

ビジネス・関数の呼出しは、次のモデルでサポートされています。

- XPI
  - エンタープライズ XPI
  - インターエンタープライズ XPI
- ERP コネクタ
  - 動的 Java コネクタ
  - Java コネクタ
  - COM コネクタ
- メッセージング・アダプタ
  - MQSeries 用アダプタ
  - MSMQ 用アダプタ
- ERP バッチ・インターフェイス
  - インターフェイス・テーブル
  - 電子データ交換(EDI)
  - テーブル変換

参照

- ビジネス関数の作成と使用については『開発ツール』ガイドの「ビジネス関数」

適切なビジネス関数の検索

ビジネス関数は、J.D. Edwards インタオペラビリティの核心部です。J.D. Edwards とのインタオペラビリティのためにカスタム・インテグレーションを構築する場合は、どのビジネス関数をどのように呼び出すかを理解する必要があります。既存の J.D. Edwards ビジネス関数をそのまま使用する方法、修正して使用する方法、カスタム・ビジネス関数を作成する方法があります。カスタム・ビジネス関数を作成する場合は、必要な機能に似た既存のビジネス関数を検索し、それをモデルとして使用することをお勧めします。既存のビジネス関数を検索するには、次の操作を試してみてください。

- オンラインのビジネス関数および API ドキュメンテーションを検討します。
- ビジネス関数ドキュメンテーションを作成します。
- 次の ERP ツールを使用します。
  - Object Management Workbench（オブジェクト管理ワークベンチ）
  - 相互参照機能
  - Autopilot Analyzer（AutoPilot アナライザ）
  - Debug Application（アプリケーションのデバッグ）

注意:

更新または ESU がビジネス関数に影響する場合は、カスタム・インテグレーションの修正が必要になることがあります。

オンラインの API およびビジネス関数ドキュメンテーションの検討

オンラインのリファレンス・ガイドには、カスタム・インテグレーションに使用できる J.D. Edwards ビジネス関数と API の詳細情報が記載されています。このガイドでは、ビジネス関数が次のようにグループ化されています。

マスター・ビジネス関数(MBF)	トランザクション全体を拡張、編集してデータベースにコミットするために必要なロジックとデータベースの呼出しを提供するビジネス関数の集合。マスター・ビジネス関数を設計すると、それを非同期で呼び出して、コード化されたエラー・メッセージを呼出し側アプリケーションに送信できます。
メジャー・ビジネス関数	日付編集ルーチンや共通多通貨機能など、多数のアプリケーションに共通の再利用可能なロジックをカプセル化するコンポーネント。
マイナー・ビジネス関数	特定のインスタンスまたは単一アプリケーションについて複合ロジックを実行するコンポーネント。ERP では、イベント・ルールで効率的に実行できない処理や、単一アプリケーションの複数箇所に必要となるロジックに、マイナー・ビジネス関数が使用されます。

参照

ビジネス関数と API については ERP のオンライン・リファレンス・ガイド

ビジネス関数ドキュメンテーションの作成

ビジネス関数ドキュメンテーションでは、個々のビジネス関数の機能とそれぞれの使用方法について説明します。すべてのビジネス関数、ビジネス関数グループ、または個々のビジネス関数に関する情報を生成できます。ビジネス関数のドキュメンテーションには、次の情報を組み込みます。

- 目的。
- パラメータ(使用するデータ構造体)。
- 個々のパラメータの説明。必要な入出力を示し、戻り値について説明します。
- 関連テーブル(アクセスするテーブル)
- 関連するビジネス関数(その関数内から呼び出されるビジネス関数)。
- 特殊取扱指示。

参照

- ビジネス関数ドキュメンテーションの作成、生成、表示については『開発ツール』ガイドの「ビジネス関数ドキュメンテーション」

モデルとしての ERP アプリケーションの使用

必要とする機能に似た機能を持つ ERP が見つかった場合は、そのアプリケーションをモデルとして使用できます。ERP の〈クロス・アプリケーション開発ツール〉メニュー(GH902)には複数のツールが用意されており、ERP アプリケーションで使用されるビジネス関数とその使用方法を判断できます。このメニューから、次のツールにアクセスできます。

- オブジェクト管理ワークベンチ
- 相互参照機能
- AutoPilot アナライザ
- アプリケーションのデバッグ

オブジェクト管理ワークベンチ

〈Object Management Workbench〉(OMW)を使用すると、ビジネス関数を検索して C コードを検討できます。

参照

- 〈オブジェクト管理ワークベンチ〉の使用方法については『開発ツール』ガイドの「オブジェクト管理ワークベンチ」

相互参照機能

相互参照機能を使用すると、ビジネス関数が使用されている各インスタンスを識別できます。〈相互参照機能〉プログラム(P980011)には、〈クロス・アプリケーション開発ツール〉メニュー(GH902)からアクセスできます。

参照

- ビジネス関数などのオブジェクトの検索については『開発ツール』ガイドの「相互参照機能」

AutoPilot Analyzer

J.D. Edwards の〈AutoPilot Analyzer（AutoPilot アナライザ）〉ツールを使用すると、ERP アプリケーションに対して実行された全ビジネス関数およびデータベース API の呼出しの入出力について、情報をキャプチャして表示できます。〈AutoPilot Analyzer〉ツールを使用するには、最初にデータをキャプチャして、このツールにインポートする必要があります。

参照

『アナライザ・ツール）』ガイドの次のトピックを参照してください。

- データのキャプチャについては「Capturing Data for OneWorld Analyzer Tool（OneWorld アナライザ・ツールのためのデータ・キャプチャ）」
- 以前にキャプチャしたデータの表示については「Importing Test Results（テスト結果のインポート）」

デバッグ・アプリケーション

ERP アプリケーションを理解する際には、もう 1 つのオプションを考慮できます。つまり、J.D. Edwards デバッガを実行することです。〈Event Rules Debugger（イベント・ルール・デバッガ）〉を実行すると、ERP アプリケーションのイベント・ルール・ビジネス関数とテーブル・イベント・ルールの情報を取得できます。Microsoft Visual C++を使用して C 言語で記述されたビジネス関数をデバッグしたり、この 2 つのツールを併用することもできます。

参照

『開発ツール』ガイドの次のトピックを参照してください。

- Event Rules Debugger の処理
- Microsoft Visual C++によるビジネス関数のデバッグ

XML および ERP ソリューション

J.D. EdwardsXML ソリューションは、的確な XML ドキュメントをサポートしています。この XML ソリューションは、受信および送信情報について UTF8 および UTF16 Unicode 標準をサポートしています。送信情報は UTF8 Unicode でサポートされ、送受信情報は UTF16 Unicode でサポートされます。J.D. Edwards には、次のような XML ソリューションが用意されています。

XMLDispatch	ERP で受信する全 XML ドキュメントと ERP の応答について、単一のエントリー・ポイントを提供します。
XML トランザクション・システム(XTS)	ERP 以外のフォーマットによる XML ドキュメントを ERP で処理できる XML ドキュメントに変換し、ERP 応答を元の XML フォーマットに変換します。
XML CallObject	J.D. Edwards ビジネス関数を呼び出すことができます。
XML トランザクション	事前定義済みのトランザクション・タイプ（JDEPOIN など）を使用して、ERP との間で情報を送信したり情報を要求できます。XML トランザクションでは、J.D. Edwards インターフェイス・テーブル機能が使用されます。
XML リスト・カーネル	ERP データベース情報をまとめて要求し、受信できます。
XML サービス・カーネル	ある ERP システムからの ERP イベントを要求し、他の ERP システムからの応答を受信できます。

XML を使用する利点は、次のとおりです。

- XML を使用するモデルはスケーラビリティを備えており、複数の接続をオープンできます。
- XML を ERP メッセージング・アダプタと共に使用して、信頼性の高い接続および確認操作を実行できます。
- XML によりビジネス関数とインターフェイス・テーブルが公開されます。
- XML を使用して、ビジネス関数の呼出しを 1 つのドキュメントにまとめることで、ネットワーク・トラフィックを減少させることができます。
- XML でセッションの作成、検証、およびトラッキングを管理できます。

インタオペラビリティ・サーバーで XML ドキュメントを作成できる場合は、インタオペラビリティ・ソリューションに XML を利用できます。XML CallObject、XML リストおよび XML トランザクション機能は、次の ERP モデルで使用できます。

- XPI
  - エンタープライズ XPI
  - インターエンタープライズ XPI
- ERP コネクタ
  - 動的 Java コネクタ
  - Java コネクタ
  - COM コネクタ
- ERP メッセージング・アダプタ
  - MQSeries 用 ERP アダプタ
  - MSMQ 用 ERP アダプタ

参照

- XML の概要については、World Wide Web Consortium (W3C)の XML ホーム・ページ http://www.w3.org/XML/

『インタオペラビリティ』ガイドの次のトピックを参照してください。

- J.D. Edwards のモデルと各モデルをサポートする機能については「インタオペラビリティの概要」
- 使用するビジネス関数の選択については「ビジネス関数の呼出し」
- XML CallObject、XML トランザクション、および XML リスト機能については「XML」
- Z イベント、リアルタイム・イベント、および XAPI イベントについては「イベント」

XML ドキュメントのフォーマット設定

XML ドキュメントを処理のために ERP に送信する場合、そのドキュメントには J.D. Edwards で定義されている XML フォーマットを使用する必要があります。ドキュメントがエンタープライズ・サーバーに到達すると、システムによりドキュメント・タイプに基づいて処理されます。すべての XML ドキュメントには、次の要素を含める必要があります。

- 次のタイプのいずれか 1 つ
  - jdeRequest タイプ
  - jdeResponse タイプ
- セッションの確立
- セッションの満期
- セッションの終了

また、必要に応じて次の要素を使用できます。

- 明示的なトランザクション
- 非明示的なトランザクション
- 準備/コミット/ロールバック

タイプ要素

タイプ要素には、jdeRequest または jdeResponse のいずれかを指定できます。これは、XML インフラストラクチャに取り込まれるすべての要求ドキュメントのルート要素です。この要素には、実行環境に関する基本情報が含まれます。jdeRequest および jdeResponse タイプ要素は、次の属性で形成されます。

Type	XML ドキュメント要求のタイプを指定します。jdeRequest タイプには、実行する操作に応じて次のいずれかを指定できます。 • Callmethod • List • Trans • xapicallmethod jdeResponse タイプは、他の ERP システムから取り込まれる XML ドキュメントを示します。jdeResponse の操作は realTimeEvent です。 **注:** xapicallmethod および realTimeEvent タイプについては、「イベント」を参照してください。

User	ユーザーの識別と検証に使用するユーザー名を指定します。
Pwd	ユーザーの識別と検証に使用するユーザー・パスワードを指定します。
Environment	システム環境を指定します。
Session	セッション ID を指定します。この属性は任意です。
Sessionidle	セッションのタイムアウト時刻を指定します。この属性は任意です。

セッションの確立

セッションを確立する必要があります。セッションの確立は、標準 jdeRequest 要素の session 属性によって処理されます。session 属性が空の文字列の場合は、新規セッションが開始されます。サーバーでは、SessionManager singleton クラスは、ユーザー名、パスワード、環境名のある session オブジェクトの新しいインスタンスを作成します。有効期限が切れる前にセッションを再利用して、セッションの初期化に伴うオーバーヘッドが発生するのを避けることができます。session 属性に、以前の要求で既に確立しているセッションのセッション ID を指定します。

```
<?xml version='1.0' ?>
<jdeRequest type='callmethod' user='steve' pwd='xyz'
environment='prod' session='' sessionidle='1800'>
...
</jdeRequest>
```

セッションの満期

セッションの満期は、標準 jdeRequest 要素の sessionidle 属性によって処理されます。この属性は、セッション作成要求で使用された場合に、このセッションがアイドル状態でいられる時間を秒単位で指定します。この時間内にセッションで何の要求も処理されなかった場合、SessionManager はセッションを終了して、すべての関連リソースを解放します。デフォルトでは、セッションのアイドル時間は 30 分です。セッションのデフォルトのアイドル時間は jde.ini ファイル内で変更できます。

```
<?xml version='1.0' ?>
<jdeRequest type='callmethod' user='steve' pwd='xyz'
environment='prod' session='' sessionidle='1800'>
...
</jdeRequest>
```

明示的なトランザクション

明示的なデータベース・トランザクションが、要素 startTransaction タグによってサポートされています。startTransaction タグでは、トランザクションを手作業でコミットするか自動的にコミットされるかを指定します。startTransaction タグ要素は空の要素です。つまり、情報はすべて属性の中にあります。

```
<?xml version='1.0' ?>
<jdeRequest type='callmethod' user='steve' pwd='xyz' environ
ment='prod' session=''>
<startTransaction trans='t1' type='manual'/>
</jdeRequest>
```

非明示的なトランザクション

トランザクション・セットの名前が trans 属性で参照されている場合、XML 要求はトランザクション・セットに組み込まれます。前に作成していないトランザクション・セットの名前を指定することによって、非明示的な開始トランザクションを要求に組み込むことができます。非明示的な開始の場合、トランザクション・セットは手作業のコミット・セットになります。

```
<?xml version='1.0' ?>
<jdeRequest type='callmethod' user='steve' pwd='xyz' environ
ment='prod' session=''>
<callMethod name='myfunc' app='P42101' trans='t1'>
<params>
<param name='CostCtr'>      1001</param>
</params>
</callMethod>
</jdeRequest>
```

準備/コミット/ロールバック

手作業のトランザクション・セットは、コミットまたはロールバックできます。2 フェーズ・コミットの一部として、コミットを準備できます。データベースに対する準備、コミットおよびロールバック要求には、endTransaction 要素が使用されます。トランザクション・セットは trans 属性で識別されます。action 属性は、トランザクション・セットに対する処理を示します。値は prepare、commit、または rollback になります。この要素は常に空の要素であり、スラッシュで示されます。

手作業でコミットするときにセッション ID を管理し、トランザクションの完了後にセッションを終了することをお勧めします。

```
<?xml version='1.0' ?>
<jdeRequest type='callmethod' user='steve' pwd='xyz' environ
ment='prod' session=''>
<endTransaction trans='t1' action='commit'/>
</jdeRequest>
```

注:

startTransaction と endTransaction が別々のドキュメントにある場合は、次のいずれかの処理が発生します。

- session 属性は、2 番目のドキュメントでは送信されません。この場合は、前のセッションの突合せに user、password、および environment が使用されます。
- 最初のドキュメントの応答セッション番号が、同じトランザクションと関連付けられたドキュメントの session 属性で送信されます。

セッションの終了

セッションを終了するには、XML ドキュメントを送信してセッションを明示的に終了します。jdeRequest 要素タグでセッションを終了するように指定する必要があります。

```
<?xml version='1.0' ?>
<jdeRequest type='callmethod' user='steve' pwd='xyz' environ
ment='prod' session=5665.931961929.454'>
<endSession/>
</jdeRequest>
```

XML ドキュメント作成に関する ERP 標準

J.D. Edwards は、XML ドキュメントについて、必要なフォーマット要素(jdeRequest または jdeResponse タイプ、セッションの確立、セッションの満期、およびセッションの終了)を使用するという要件に加えて、業界標準とは異なる標準を設けています。

小数点とカンマの区切文字

J.D. Edwards では、小数点と 1,000 の位に XML 業界標準とは異なる区切文字を使用しています。この 2 つの区切文字は、使用するプロファイル設定、jde.ini 設定、またはコンピュータの地域設定に依存しません。ERP とインターフェイスする XML ドキュメントを記述する際には、小数点区切文字として常に小数点記号「.」(ドット)を使用し、1,000 の位の区切文字として常にカンマ「,」を使用する必要があります。区切文字標準の目的は、インタオペラビリティ方針の一貫性を保ち、データの破損を防ぐことです。

日付の使用

XML ファンデーションのコンポーネントごとに、異なる形式コードと API を使用して次の日付がフォーマットされます。

- To XML 日付
- From XML 日付
- To JDEDATE
- From JDEDATE

次の表に、J.D. Edwards でサポートしている各 XML コンポーネントで使用されるフォーマットを示します。

	受信		送信	
コンポーネント	**形式**	**結果**	**形式**	**結果**
XMLCallObject	F	YYYYMD	ESOSA	YYYY/MM/DD
XML トランザクション	F*	ユーザー設定	ESOSA	YYYY/MM/DD
XMLList	B*	ユーザー設定	NULL	ユーザー設定

* コンポーネントでは形式コードが無視されます。

XML のシステム環境の構成

ERP で XML を使用する前に、ERP エンタープライズ・システム上で ICU_DATA システム環境変数が適切に定義されていることを確認する必要があります。ICU_DATA 変数が適切に定義されていない場合は、次のエラー・メッセージが表示されます。

> The default Unicode converter could not be found within the jdenet_n.log on the OneWorld Enterprise Server.(OneWorld エンタープライズ・サーバーの jdenet_n.log に、デフォルトの Unicode コンバータが見つかりません。)

ERP の場合、ICU 変換テーブル icu_data.dat は一般に system/locale/xml にあります。ここでは、各種プラットフォーム用の設定について説明します。使用中のプラットフォームに該当する設定を使用してください。

UNIX

UNIX システムの場合、ICU_DATA のパスは ICU_DATA 環境変数に基づいています。UNIX の ERP ユーザー・ログイン・スクリプトでは、ICU_DATA 環境変数が ICU リソース・ファイル incudata.dat のディレクトリに設定されます。ユーザー・ログイン・スクリプトで ICU_DATA 環境変数が設定されない場合は、ICU_DATA 変数に後続スラッシュを付けて次のように定義する必要があります。

```
Export ICU_DATA=$SYSTEM/locale/xml/
```

この場合、$SYSTEM は ERP インストール・ディレクトリを表します。

AS/400

AS/400 システムの場合、ICU_DATA のパスはシステムで ICU 1.6 変換関数が最初に呼び出されるときに設定されます。システム・ライブラリ内のデータ領域 BUILD_VER で System Directory 設定が検索されます。次に例を示します。

System Directory:B9_S

BUILD_VER で指定されているパスの末尾に locale/xml が追加されてから、ICU_DATA のパスとして使用されます。BUILD_VER データ領域がシステム・ディレクトリの設定を反映するように正しく設定されているかどうかを確認する必要があります。

WIN32

WIN32 システムの場合、ICU_DATA のパスは ICU 1.6 変換関数が最初に呼び出されるときに設定されます。WIN32 では、次のロジックが使用されます。

1. 環境変数 JDE_B9_ICU_DATA が検索されます。この環境変数が見つかった場合は、それが変換ファイルのパスとなります。
2. jde.ini ファイル内で次のセクションが検索されます。

   [XML]

   ICUPath=<<install>>/system/locale/xml

   ICUPath 設定が見つかった場合は、それが変換ファイルのパスとなります。
3. jde.ini ファイル内で ICUPath 設定が見つからない場合は、ICU_Path が次のようにデフォルト設定されます。

   EXECUTABLE_DIRECTORY/./system/locale/xml

   The EXECUTABLE_DIRECTORY must be <<install>>/system/bin32.

上記のロジックにより、通常、JDE_B9_ICU_DATA 環境変数または jde.ini ファイルを設定する必要はありません。jde.ini の ICUPath を設定する必要があるのは、icudata.dat が system/locale/xml とは異なる位置にある場合のみです。

注:

ERP クライアント・インストールでは、環境変数 JDE_B9_ICU_DATA が設定されます。

XML 変換サービス

J.D. Edwards の XML 変換システム(XTS)では、拡張可能スタイルシート言語(eXtensible Stylesheet Language：XSL)を使用して、ERP 以外のフォーマットを持つ XML ドキュメントが、ERP に必要な XML フォーマットに変換されます。また、XTS により、ERP 応答の XML ドキュメントが元の要求の XML フォーマットに変換されます。

XTS は、ERP カーネル・プロセスとして実行されるマルチスレッド化 Java プロセスです。システムの起動時には、XTS カーネル・ライブラリにより Java 仮想マシン(JVM)がロードされます。JVM のロード後に、サーバー・プロキシが起動されます。サーバー上で使用可能な JVM がない場合は、インストールする必要があります。J.D. Edwards は、Java JVM バージョン 1.3 以上をサポートしています。

XTS は、ERP がサポートしている全プラットフォーム上で使用可能です。

XTS 処理

ERP の XML ディスパッチ・カーネルは、認識されない XML ドキュメントを受信すると、それを変換のために XTS に送信します。XTS は XSL を読み取り、ドキュメントを ERP 互換フォーマットに変換してから、ERP で処理するために XML ディスパッチ・カーネルに送信します。ERP 応答が XML ディスパッチに着信すると、XML ディスパッチはドキュメントを ERP の XML フォーマットから変換する必要があることを記憶してから、そのドキュメントを変換のために XTS に送信します。XTS は ERP の XML ドキュメントを元の XML フォーマットに変換してから、ユーザーに配布できるように XML ディスパッチに送信します。

ERP に着信する XML ドキュメントはすべて、ネイティブ XML フォーマットであることが必要です。ネイティブ XML フォーマットとは、本書で説明する J.D. Edwards 定義の XML フォーマットです。ERP のカーネル・プロセス（XML CallObject、XML trans、XML リストなど）が処理できるのは、ネイティブ・フォーマットの XML ドキュメントのみです。J.D. Edwards には、XTS ソリューションの一部として、ERP 以外の XML ドキュメントが変換可能かどうかを判別するセレクタが用意されています。セレクタとは、XML ドキュメントを調べて認識できるかどうかを確認するコードです。セレクタで XML ドキュメントが認識される場合は、そのドキュメントを変換用に用意されているスタイルシートに関連付けることができます。J.D. Edwards 提供のセレクタには、バージョン 1 の XML フォーマットを ERP のネイティブ XML フォーマットに変換する機能があります。バージョン 1 の XML フォーマットは J.D. Edwards 定義の XML フォーマットですが、ツールとして使用しやすいように修正が加えられています。ネイティブ XML フォーマットではパラメータ名で始まるフィールド名が使用されますが、バージョン 1 の XML フォーマットではフィールド名のみが使用されます。

例：ERP のネイティブ XML フォーマット

次のサンプルに、ERP のネイティブ XML フォーマットを示します。

```
<?xml version="1.0" encoding="UTF-8" ?>
<jdeRequest type="callmethod" user="JDE" pwd="JDE" environment="PRD733" session="">
 <callMethod name="GetLocalComputerId" app="MSMQ" runOnError="no">
  <params>
   <param name="szMachineKey" id="2" />
  </params>
  <onError abort="yes" />
 </callMethod>
```

```
- <callMethod name="F4211FSBeginDoc" app="MSMQ" runOnError="no">
   <params>
    <param name="mnCMJobNumber" id="1" />
    <param name="cCMDocAction">A</param>
    <param name="cCMProcessEdits">1</param>
    <param name="szCMComputerID" idref="2" />
    <param name="cCMUpdateWriteToWF">2</param>
    <param name="szCMProgramID">MSMQ</param>
    <param name="szCMVersion">MSMQ</param>
    <param name="szOrderType">SQ</param>
    <param name="szBusinessUnit">M30</param>
    <param name="mnAddressNumber">4242</param>
    <param name="szReference">2</param>
    <param name="cApplyFreightYN">Y</param>
    <param name="szCurrencyCode">CAD</param>
    <param name="cWKSourceOfData" />
    <param name="cWKProcMode">1</param>
    <param name="mnWKSuppressProcess">0</param>
   </params>
   <onError abort="yes">
    <callMethod name="F4211ClearWorkFile" app="MSMQ" runOnError="yes">
     <params>
      <param name="mnJobNo" idref="1" />
      <param name="szComputerID" idref="2" />
      <param name="mnFromLineNo">0</param>
      <param name="mnThruLineNo">0</param>
      <param name="cClearHeaderWF">2</param>
      <param name="cClearDetailWF">2</param>
      <param name="szProgramID">MSMQ</param>
      <param name="szCMVersion">ZJDE0001</param>
     </params>
    </callMethod>
   </onError>
  </callMethod>
  <callMethod name="F4211FSEditLine" app="MSMQ" runOnError="yes">
   <params>
    <param name="mnCMJobNo" idref="1" />
    <param name="cCMLineAction">A</param>
    <param name="cCMProcessEdits">1</param>
    <param name="cCMWriteToWFFlag">2</param>
    <param name="szCMComputerID" idref="2" />
    <param name="mnLineNo">1</param>
    <param name="szItemNo">1001</param>
    <param name="mnQtyOrdered">5</param>
    <param name="cSalesTaxableYN">N</param>
    <param name="szTransactionUOM">EA</param>
    <param name="szCMProgramID">1</param>
    <param name="szCMVersion">ZJDE0001</param>
    <param name="cWKSourceOfData" />
   </params>
   <onError abort="no" />
  </callMethod>
  <callMethod name="F4211FSEndDoc" app="MSMQ" runOnError="no">
   <params>
    <param name="mnCMJobNo" idref="1" />
```

```
    <param name="szCMComputerID" idref="2" />
    <param name="szCMProgramID">MSMQ</param>
    <param name="szCMVersion">ZJDE0001</param>
    <param name="cCMUseWorkFiles">2</param>
    <param name="mnSalesOrderNo" id="3" />
    <param name="szKeyCompany" id="4" />
    <param name="mnOrderTotal" id="5" />
   </params>
   <onError abort="no">
    <callMethod name="F4211ClearWorkFile" app="MSMQ" runOnError="yes">
     <params>
      <param name="mnJobNo" idref="1" />
      <param name="szComputerID" idref="2" />
      <param name="mnFromLineNo">0</param>
      <param name="mnThruLineNo">0</param>
      <param name="cClearHeaderWF">2</param>
      <param name="cClearDetailWF">2</param>
      <param name="szProgramID">MSMQ</param>
      <param name="szCMVersion">ZJDE0001</param>
     </params>
    </callMethod>
   </onError>
  </callMethod>
  <returnParams failureDestination="error" runOnError="yes" successDestination="success">
   <param name="mnOrderNo" idref="3" />
   <param name="szOrderCo" idref="4" />
   <param name="mnWKOrderTotal" idref="5" />
  </returnParams>
  <onError abort="yes">
   <callMethod name="F4211ClearWorkFile" app="MSMQ" runOnError="yes">
    <params>
     <param name="mnJobNo" idref="1" />
     <param name="szComputerID" idref="2" />
     <param name="mnFromLineNo">0</param>
     <param name="mnThruLineNo">0</param>
     <param name="cClearHeaderWF">2</param>
     <param name="cClearDetailWF">2</param>
     <param name="szProgramID">MSMQ</param>
     <param name="szCMVersion">ZJDE0001</param>
    </params>
   </callMethod>
  </onError>
</jdeRequest>
```

例：ERP のバージョン 1 の XML フォーマット

次のサンプルに、バージョン 1 の XML フォーマットを示します。

```
<?xml version="1.0" ?>
<intBPAPI>
 <dsControl>
  <dsLogin>
```

```
    <User>JDESVR</User>
    <Password>JDESVR</Password>
    <Environment>ADEVNIS2</Environment>
    <Session />
  </dsLogin>
  <dsAPI>
    <Noun>jdeSalesOrder</Noun>
    <Verb>Create</Verb>
    <Version>1.1</Version>
  </dsAPI>
  <dsTranslation>
    <InMap />
    <OutMap />
  </dsTranslation>
</dsControl>
<dsData>
  <callMethod_GetLocalComputerId app="NetComm" runOnError="no">
    <szMachineKey id="2" />
    <onError_GetLocalComputerId abort="yes" />
  </callMethod_GetLocalComputerId>
<callMethod_F4211FSBeginDoc app="NetComm" runOnError="no">
  <mnCMJobNumber id="1" />
  <cCMDocAction>A</cCMDocAction>
  <cCMProcessEdits>1</cCMProcessEdits>
  <szCMComputerID idref="2" />
  <cCMUpdateWriteToWF>2</cCMUpdateWriteToWF>
  <szCMProgramID>NetComm</szCMProgramID>
  <szCMVersion>NetComm</szCMVersion>
  <szOrderType>SQ</szOrderType>
  <szBusinessUnit>M30</szBusinessUnit>
  <mnAddressNumber>4242</mnAddressNumber>
  <szReference>2</szReference>
  <cApplyFreightYN>Y</cApplyFreightYN>
  <szCurrencyCode>CAD</szCurrencyCode>
  <cWKSourceOfData />
  <cWKProcMode>1</cWKProcMode>
  <mnWKSuppressProcess>0</mnWKSuppressProcess>
  <onError_F4211FSBeginDoc abort="yes">
    <callMethod_F4211ClearWorkFile app="NetComm" runOnError="yes">
      <mnJobNo idref="1" />
      <szComputerID idref="2" />
      <mnFromLineNo>0</mnFromLineNo>
      <mnThruLineNo>0</mnThruLineNo>
      <cClearHeaderWF>2</cClearHeaderWF>
      <cClearDetailWF>2</cClearDetailWF>
      <szProgramID>NetComm</szProgramID>
      <szCMVersion>ZJDE0001</szCMVersion>
    </callMethod_F4211ClearWorkFile>
  </onError_F4211FSBeginDoc>
</callMethod_F4211FSBeginDoc>
  <callMethod_F4211FSEditLine app="NetComm" runOnError="yes">
    <mnCMJobNo idref="1" />
    <cCMLineAction>A</cCMLineAction>
    <cCMProcessEdits>1</cCMProcessEdits>
    <cCMWriteToWFFlag>2</cCMWriteToWFFlag>
```

```
    <szCMComputerID idref="2" />
    <mnLineNo>1</mnLineNo>
    <szItemNo>1001</szItemNo>
    <mnQtyOrdered>5</mnQtyOrdered>
    <cSalesTaxableYN>N</cSalesTaxableYN>
    <szTransactionUOM>EA</szTransactionUOM>
    <szCMProgramID>1</szCMProgramID>
    <szCMVersion>ZJDE0001</szCMVersion>
    <cWKSourceOfData />
    <onError_F4211FSEditLine abort="no" />
   </callMethod_F4211FSEditLine>
   <callMethod_F4211FSEndDoc app="NetComm" runOnError="no">
    <mnCMJobNo idref="1" />
    <szCMComputerID idref="2" />
    <szCMProgramID>NetComm</szCMProgramID>
    <szCMVersion>ZJDE0001</szCMVersion>
    <cCMUseWorkFiles>2</cCMUseWorkFiles>
    <mnSalesOrderNo id="3" />
    <szKeyCompany id="4" />
    <mnOrderTotal id="5" />
    <onError_F4211FSEndDoc abort="no">
     <callMethod_F4211ClearWorkFile app="NetComm" runOnError="yes">
      <mnJobNo idref="1" />
      <szComputerID idref="2" />
      <mnFromLineNo>0</mnFromLineNo>
      <mnThruLineNo>0</mnThruLineNo>
      <cClearHeaderWF>2</cClearHeaderWF>
      <cClearDetailWF>2</cClearDetailWF>
      <szProgramID>NetComm</szProgramID>
      <szCMVersion>ZJDE0001</szCMVersion>
     </callMethod_F4211ClearWorkFile>
    </onError_F4211FSEndDoc>
   </callMethod_F4211FSEndDoc>
   <returnParams failureDestination="error" successDestination="success" runOnError="yes">
    <mnOrderNo idref="3" />
    <szOrderCo idref="4" />
    <mnWKOrderTotal idref="5" />
   </returnParams>
   <onError abort="yes">
    <callMethod_F4211ClearWorkFile app="NetComm" runOnError="yes">
     <mnJobNo idref="1" />
     <szComputerID idref="2" />
     <mnFromLineNo>0</mnFromLineNo>
     <mnThruLineNo>0</mnThruLineNo>
     <cClearHeaderWF>2</cClearHeaderWF>
     <cClearDetailWF>2</cClearDetailWF>
     <szProgramID>NetComm</szProgramID>
     <szCMVersion>ZJDE0001</szCMVersion>
    </callMethod_F4211ClearWorkFile>
   </onError>
  </dsData>
</intBPAPI>
```

カスタム・セレクタの作成

XML フォーマットを ERP のネイティブ XML フォーマットに変換するセレクタを作成できます。カスタム・セレクタを記述する場合は、要求と応答の拡張可能スタイルシート言語変換(eXtensible Stylesheet Language Transformation：XSLT)ドキュメントを含めてください。

Java ファイル内では、テンプレートを選択するために 2 つの API が使用されます。要求ドキュメントに該当する XSLT ドキュメントをフェッチするには、ブールの fetchTemplates API を使用します。パブリックのブール fetchTemplates は、IXTSMTemplateSelector.TemplateFetchException、XTSXMLParseException を発生させます。次のサンプルに、この API の使用方法を示します。

```
fetchTemplates(XTSDocument  inXML,  IXTSMSelectionInfo  info)
```

応答ドキュメントに該当する XSLT ドキュメントをフェッチするには、パブリック void の fetchTemplates を使用します。パブリック void の fetchTemplates は、IXTSMTemplateSelector.TemplateFetchException を発生させます。

```
fetchTemplates(IXTSMSelectionInfo  info)
```

J.D. Edwards によるセレクタの記述方法を示すサンプル・コードについては、下記を参照してください。

注：

カスタム・セレクタが ClassPath 内でアクセス可能になっていることを確認してください。

XTS の API

- IXTSMTemplateSelector
- IXTSMTemplateSelector.TemplateFetchException

この 2 つの API については、オンラインのリファレンス・ガイドを参照してください。

例：セレクタの作成

次のコードは、バージョン 1 の XML セレクタを作成するために J.D. Edwards が記述したものです。

```
File: XTSMJDETemplateSelector.java
//
////////////////////////////////////////////////////////////////////////////
//
// Copyright (c) 2001 J.D. Edwards World Source Company
//
// This unpublished material is proprietary to J.D. Edwards World Source
// Company. All rights reserved.  The methods and techniques described
// herein are considered trade secrets and/or confidential.  Reproduction
// or distribution, in whole or in part, is forbidden except by express
// written permission of J.D. Edwards World source Company.
//
////////////////////////////////////////////////////////////////////////////
```

```
package com.jdedwards.xts.xtsm;

import com.jdedwards.xts.xtsr.IXTSRepository;
import com.jdedwards.xts.xtsr.IXTSRKey;
import com.jdedwards.xts.xtsr.XTSRException;
import com.jdedwards.xts.xtsr.XTSRInvalidKeyStringException;
import com.jdedwards.xts.xtsr.XTSRInvalidKeyFieldException;
import com.jdedwards.xts.xtsr.XTSRKeyNotFoundException;

import com.jdedwards.xts.XTSDocument;
import com.jdedwards.xts.XTSFactory;
import com.jdedwards.xts.XTSLog;
import com.jdedwards.xts.XTSConfigurationException;
import com.jdedwards.xts.XTSXMLParseException;

import com.jdedwards.xts.xtsm.IXTSMTemplateSelector;

import com.jdedwards.xts.xtse.IXTSEngine;
import com.jdedwards.xts.xtse.IXTSECompiledProcessor;

import java.util.List;
import org.w3c.dom.*;


/**
 * This class is the JDEdwards Template Selector. It will recognize
 * J.D. Edwards standard XML documents, and return the appropriate XSL
 * stylesheets necessary for transformation.
 */
public class XTSMJDETemplateSelector implements IXTSMTemplateSelector
{
  /** Class constructor. */
  public XTSMJDETemplateSelector()
  {
    XTSLog.trace("XTSMJDETemplateSelector()", 3);

    // get repository reference
    XTSFactory factory = XTSFactory.getInstance();
    m_repository = factory.createXTSRepository();
  }

  /**
   * Fetch the appropriate XSLT document(s) and/or IXTSECompiledProcessors as
   * indicated by the TPT stored in the <code>info</code> parameter.
   * @param info - Selection Info which contains TPI and should be modified
   * by the selector to specify transformation information.
   * @exception IXTSMTemplateSelector.TemplateFetchException - thrown
   * if an error occurs when extracting information from the
   * inclement.
   */
  public void fetchTemplates(IXTSMSelectionInfo info)
        throws IXTSMTemplateSelector.TemplateFetchException
  {
    XTSLog.trace("XTSMJDETemplateSelector.fetchTemplates(XTSMSelectionResult)", 3);
```

```
    NodeList nodes = info.getTPIElement().getElementsByTagName(JDE_TS_XTSR_KEY);
    int numNodes = nodes.getLength();
    for(int i = 0; i < numNodes; i++)
    {
      // extract key info & create a key
      IXTSRKey key = createKeyFromNode((Element)nodes.item(i));

      // fetch the doc and add it to the list
      try
      {
        info.getXSLList().add(m_repository.fetch(key));
      }
      catch (XTSRKeyNotFoundException e)
      {
        throw new IXTSMTemplateSelector.TemplateFetchException(
            "Selected XTSRKey not found in repository: "
            + JDE_TS_XTSR_KEY);
      }
      catch (XTSRException e)
      {
        throw new IXTSMTemplateSelector.TemplateFetchException(
            "Unable to fetch the XSL document specified within '"
            + JDE_TS_XTSR_KEY +
            "' from the XTSRepository");
      }
    }
  }

  /**
   * Fetch the appropriate XSLT document(s)/compiled processors for the given
   * document.
   * @param inXML - the XTSDocument to try to recognize.
   * @param info - Selection Info object to be modified by selector to
   * indicate transformation information.
   * @return - <code>true</code> if the selector has recognized the document
   * and specified the appropriate selection info via <code>info</code>,
   * <code>false</code> otherwise.
   * @exception TemplateFetchException - thrown when an error occurs when
   * trying to recognize the DOM.
   * @exception XTSXMLParseException - thrown if <inXML> could not be parsed.
   */
  public boolean fetchTemplates(XTSDocument inXML,
                                IXTSMSelectionInfo info)
    throws IXTSMTemplateSelector.TemplateFetchException,
           XTSXMLParseException
  {
    XTSLog.trace("XTSMJDETemplateSelector.fetchTemplates(Document, Element)", 3);

    boolean recognized = false;
    Document inDOM = inXML.getDOM();
    // see if a XTSR key is specified within the document:
    NodeList nodeList = inDOM.getElementsByTagName(JDE_XTSR_KEY);

    if (nodeList.getLength() > 0)
```

```
    {
      try
      {
        // extract key info & create a key
        IXTSRKey key = createKeyFromNode((Element)nodeList.item(0));

        // add transformation path information to outElement
        createNodeChildFromKey(info.getTPIElement(), key);

        // fetch the doc and add it to the list
        info.getXSLList().add(m_repository.fetch(key));

        info.setResultXML(true);
        info.setPathInfoStored(false);

        recognized = true;
      }
      catch (XTSRException e)
      {
        throw new IXTSMTemplateSelector.TemplateFetchException(
                "Unable to fetch the XSL document specified within '"
                + JDE_XTSR_KEY +
                "' from the XTSRepository");
      }
      catch (XTSRKeyNotFoundException e)
      {
        throw new IXTSMTemplateSelector.TemplateFetchException(
                "Key specified in TPI not found in repository"
                + JDE_XTSR_KEY);
      }
    }
    else // no XTSR key, so look for JDE information:
    {
      nodeList = inDOM.getElementsByTagName(JDE_INT_BPAPI);
      if (nodeList.getLength() != 0)
      {
        // add transformation path information to outElement
        createNodeChildFromKey(info.getTPIElement(), getVersion1toNativeKey());

        // fetch the doc and add it to the list
        info.getXSLList().add(getVersion1toNativeXSL());

        info.setResultXML(true);
        info.setPathInfoStored(true);

        recognized = true;
      }
    }

    return recognized;
  }

  /**
   * Extracts XTSRKey information from the given node, and creates an
   * instance of IXTSRKey based on that information.
```

```
 * @return – the new IXTSRKey.
 * @param element – Element that contains the key information.
 * @exception XTSMUnrecognizedElementException – thrown if the Element
 * format is unrecognized.
 */
protected IXTSRKey createKeyFromNode(Element element)
        throws XTSMUnrecognizedElementException
{
   XTSLog.trace("XTSMJDETemplateSelector.createKeyFromNode(Element)", 4);

   IXTSRKey key = null;
   boolean request = false;
   boolean response = false;
   if (element.getNodeName().equals(JDE_XTSR_KEY))
   {
      request = true;
   }
   else if (element.getNodeName().equals(JDE_TS_XTSR_KEY))
   {
      response = true;
   }
   if (request || response)
   {
      key = m_repository.createKey();
      try
      {
         String keyString = element.getAttribute(JDE_XTSR_KEY_ATTRIBUTE);
         key.setFieldsFromString(keyString);
         if (key.getFieldValue(SUBTYPE_FIELD).length() == 0)
         {
            if (request)
            {
               key.setFieldValue(SUBTYPE_FIELD, SUBTYPE_REQUEST);
            }
            else
            {
               key.setFieldValue(SUBTYPE_FIELD, SUBTYPE_RESPONSE);
            }
         }
      }
      catch (XTSRInvalidKeyStringException e)
      {
         throw new XTSMUnrecognizedElementException(
                  "Specified '" + JDE_XTSR_KEY +
                  "' element format is invalid for this XTSRepository");
      }
      catch (XTSRInvalidKeyFieldException e)
      {
         throw new XTSConfigurationException(
                  "Specified '" + SUBTYPE_FIELD +
                  "' field name not supported by repository key");
      }
   }
   return key;
}
```

```
/**
 * Creates a node that contains the key fields values and appends it to
 * the given parentNode.
 * @param parentNode - Node to which the key information should be
 * appended.
 * @param key - Key information to store in the node.*/
protected void createNodeChildFromKey(Node parentNode, IXTSRKey key)
{
   XTSLog.trace("XTSMJDETemplateSelector.createKeyFromNode(Node,IXTSRKey)", 4);

   try
   {
      IXTSRKey keyClone = key.getRepository().createKey();
      keyClone.setFieldsFromString(key.getFieldsString());

      // we don't want to store the sub type, so we will clear it here:
      keyClone.setFieldValue(SUBTYPE_FIELD, "");

      // create new node and append it to the provided element:
      Element element = (Element)parentNode.getOwnerDocument().createElement(JDE_TS_XTSR_KEY);
      element.setAttribute(JDE_XTSR_KEY_ATTRIBUTE, keyClone.getFieldsString());
      parentNode.appendChild(element);
   }
   catch (XTSRInvalidKeyStringException e)
   {
      XTSLog.log("Unexpected ");
      XTSLog.log(e);
      throw new RuntimeException("Unexpected Exception: " + e.toString());
   }

}

/**
 * Returns the key of the stylesheet to use in converting JDE version 1
 * documents into JDE native documents.
 * @return - The key for the XSL stylesheet.
 */
protected IXTSRKey getVersion1toNativeKey()
{
   XTSLog.trace("XTSMJDETemplateSelector.getVersion1toNativeKey()", 5);
   if (null == m_version1ToNativeKey)
   {
      try
      {
         // create standard xsl XTSRKey:
         m_version1ToNativeKey = m_repository.createKey();
         m_version1ToNativeKey.setFieldsFromString(V1_TO_NATIVE_KEY);
      }
      catch (XTSRInvalidKeyStringException e)
      {
         String error = "XTSRKey necessary for JDE template selection is invalid: "
               + V1_TO_NATIVE_KEY;
         XTSLog.log(error);
         XTSLog.log(e);
```

```
            throw new XTSConfigurationException(error);
        }
    }
    return m_version1ToNativeKey;
}

/**
 * Returns the XTSDocument which contains the XSL stylesheet for converting
 * JDE version 1 documents into JDE native documents.
 * @return - XTSDocument containing the XSL stylesheet.
 */
protected IXTSECompiledProcessor getVersion1toNativeXSL()
{
    XTSLog.trace("XTSMJDETemplateSelector.getVersion1toNativeXSL()", 5);
    if (null == m_version1ToNativeXSL)
    {
        XTSDocument xsl = null;
        Try
        {
            xsl = m_repository.fetch(getVersion1toNativeKey());
            IXTSEngine engine = XTSFactory.getInstance().createXTSEngine();
            m_version1ToNativeXSL = engine.createCompiledProcessor(xsl);
        }
        catch (XTSRException e)
        {
            String error = "Unable to fetch selected template from the repository: ";
            XTSLog.log(error);
            XTSLog.log(e);
            throw new XTSConfigurationException(error + e.toString());
        }
        catch (XTSRKeyNotFoundException knfe)
        {
            String error = "Selected template XTSRKey not found in repository:";
            XTSLog.log(error);
            XTSLog.log(knfe);
            throw new XTSConfigurationException(error + knfe.toString());
        }
        catch (XTSXMLParseException pe)
        {
            String error = "Invalid XSL document in repository";
            XTSLog.log(error);
            XTSLog.log(pe);
            throw new XTSConfigurationException(error + pe.toString());
        }
    }
    return m_version1ToNativeXSL;
}

/** Referenct to the XTSRepository */
private IXTSRepository      m_repository          = null;

/** Key for converting JDE version 1 documents to JDE native documents. */
private IXTSRKey            m_version1ToNativeKey  = null;

/** Compiled XSL Stylesheet for converting JDE version 1 docs to
```

```
 *  JDE native docs. */
private IXTSECompiledProcessor m_version1ToNativeXSL  = null;

/** Field Value for the XTSRKey that indicates the document is an XSL doc */
private static final String DOC_TYPE_XSL          = "XSL";

/** Element name that indicates the DOM is a JDEdwards Version 1 document */
private static final String JDE_INT_BPAPI         = "intBPAPI";

/** Element name that indicates the DOM is a request and not a response or
 *  error. */
private static final String JDE_REQUEST           = "jdeRequest";

/** Element name that indicates the DOM is a response */
private static final String JDE_RESPONSE          = "jdeResponse";

/** Element name that specifies an XTSRKey to use in transforming the
 *  document. */
private static final String JDE_XTSR_KEY          = "jdeXTSRKey";

/** The attribute of the <code>JDE_XTSR_KEY</code> element which stores
 *  the XTSRKey string value */
private static final String JDE_XTSR_KEY_ATTRIBUTE = "key";

/** XTSRKey field name which specifies the sub-type of the XML document.
 *  Normal values for the sub-type are defined by
 *  <code>SUBTYPE_REQUEST</code> and <code>SUBTYPE_RESPONSE</code> */
private static final String SUBTYPE_FIELD         = "SUB_TYPE";

/** XTSRKey field name which specifies the type of the XML document.
 *  The normal value is defined by <code>DOC_TYPE_XSL</code> */
private static final String FIELD_TYPE            = "TYPE";

/** XTSRKey field name which specifies the format (or owner) of the XML
 *  document.  The normal value recognized by this selector is 'JDE' */
private static final String FIELD_FORMAT          = "FORMAT";

/** XTSRKey field name which specifies the particular transformation which
 *  the XSL document will perform.  This selector uses 'V1_NATIVE' for
 *  transformations between JDEdwards Version 1 XML documents and J.D. Edwards
 *  native version documents. */
private static final String FIELD_ID              = "ID";

/** The string representation of the XTSRKey for the XSL document to format
 *  JDEdwards version 1 request documents into JDEdwards native request
 *  documents. */
private static final String V1_TO_NATIVE_KEY      = "XSL-JDE-V1_NATIVE-REQUEST";

/** XTSRKey field <code>SUBTYPE_FIELD</code> value that indicates the XSL
 *  document will transform jdeRequest documents. */
private static final String SUBTYPE_REQUEST       = "REQUEST";

/** XTSRKey field <code>SUBTYPE_FIELD</code> value that indicates the XSL
 *  document will transform jdeResponse documents. */
private static final String SUBTYPE_RESPONSE      = "RESPONSE";
```

```
  /** Element name stored within the Transformation Path Information (TPI)
   *  which specifies the XTSRKey used to transform the document. */
  private static final String JDE_TS_XTSR_KEY        = "XTSJDETemplateKey";

  private static class XTSMUnrecognizedElementException
        extends IXTSMTemplateSelector.TemplateFetchException
  {
    public XTSMUnrecognizedElementException(String text)
    {
      super(text);
    }
  }
}
```

XTS の jde.ini ファイルの構成

XTS カーネルは、サーバーの jde.ini ファイル内で定義する必要があります。構成ファイル名は、JVM の config_file システム変数から取得され、これらのプロパティ設定は jde.ini 以外の構成ファイルの一部となっています。XTS カーネルを定義することを除き、jde.ini ファイルに特別な設定は不要です。

カーネル設定は次のとおりです。

```
[JDENET_KERNEL_DEF23]
    krnlName=JDEXTS KERNEL
    dispatchDLLName=xtskrnl.dll
    dispatchDLLFunction=_JDEK_DispatchXTSMessage@28
    maxNumberOfProcesses=1
    numberOfAutoStartProcesses=0
```

他のプラットフォームのさまざまな.dll 拡張子は、次のテーブルを参照してください。

	dispatchDLLName	dispatchDLLFunction
AS400	XTSKRNL	JDEK_DispatchXTS
HP9000B	libxtskrnl.sl	JDEK_DispatchXTS
SUN または RS6000	libxtskrnl.so	JDEK_DispatchXTS

その他の jde.ini ファイル設定は次のとおりです。

```
[JDE_CG]
CLASSPATH=<install-path>¥system¥Classes¥xalan.jar;<install-
path>¥system¥Classes ¥xerces.jar;<install-path>¥system¥Classes¥_Kernel.jar;<install-
path>¥system¥Classes¥XTS.jar; <install-path>¥system¥Classes¥log4j.jar;<install-
path>¥system¥Classes
```

注:

CLASSPATH を更新し、[XTS]セクションの XTSTemplaceSelector2 に指定されているカスタム・クラス・ファイルを組み込んでください。

CLASSPATH には$SYSTEM を使用しないでください。システム・ディレクトリのフル・パスを指定する必要があります。

[JDENET]
maxKernelRanges=24.

注:

XTS カーネルを実行するには、maxKernelRanges を 23 以上に設定する必要があります。

[XTSRepository]
XSL-JDE-V1_NATIVE-REQUEST=ml.xsl
XSL-JDE-V1_NATIVE-RESPONSE=lm.xsl

注:

最初の設定は J.D. Edwards のデフォルト XSL で、要求ドキュメントがバージョン 1 フォーマットからネイティブ・フォーマットに変換されます。2 番目の設定は J.D. Edwards のデフォルト XSL で、応答ドキュメントがネイティブ・フォーマットからバージョン 1 フォーマットに変換されます。

セレクタで XSL を検索してアクセスできる位置であれば、XSL ファイルをこの位置または他の任意の位置に置くことができます。この位置に XSL ファイルを追加したい場合は、次の命名規則に従ってください。この場合、Filename は XSL ドキュメント名です。

XSL-JDE-Filename-REQUEST=

XSL-JDE-Filename-RESPONSE=

[XTS]
XTSTemplateSelector1=com.jdedwards.xts.xtsm.XTSMJDETemplateSelector
XTSTraceLevel=2

注:

XTSTemplaceSelector1 設定は J.D. Edwards のデフォルト・テンプレート・セレクタで、バージョン 1 フォーマットとネイティブ・フォーマットの間で変換する XSL を提供します。

このセクションにカスタム・テンプレート・セレクタを追加できます。たとえば、テンプレート・セレクタ設定を次のように定義できます。

XTSTemplateSelector2=com.customer.CustomTemplateSelector

XTSTraceLevel=2 設定では、XTS のロギング・レベルを定義します。

XML ディスパッチ

XML ディスパッチは、ERP カーネル・プロセスとして実行される XML ベースのインタオペラビリティ機能です。この XML ディスパッチ・カーネルは、すべての XML ドキュメントに関する ERP の一元的なエントリ・ポイントとなります。XML ディスパッチは ERP に着信する XML ドキュメントの種類を識別し、そのドキュメントを処理のために該当するカーネルに送信します。XML ディスパッチで認識できないドキュメントは XTS に送信され、XTS がそのドキュメントを認識してネイティブ ERP フォーマットに変換します。XTS により変換された後、ドキュメントは XML ディスパッチに送り返され、そこから処理のために該当するカーネルに送信されます。発信ドキュメントの場合、XML ディスパッチは要求ドキュメントが ERP ネイティブ・フォーマットに変換されたかどうかを記憶できます。着信要求が変換されていた場合、発信応答ドキュメントは XTS に送信され、XTS でネイティブ ERP フォーマットから元の要求のフォーマットに変換されます。XTS で変換された後、ドキュメントは XML ディスパッチに送信され、入力者に配布されます。

XML ディスパッチ・カーネルには、XML ドキュメントのルーティング機能とロード・バランシング機能があります。たとえば、一度に多数の XML CallObject メッセージ・タイプが着信すると、XML ディスパッチは新規の CallObject カーネルをインスタンス化しようとします。カーネルが使用可能なインスタンスの数は、jde.ini ファイル内で設定します。たとえば、CallObject カーネルのインスタンス数を 5 に設定した場合、複数のドキュメントが ERP に着信すると、XML ディスパッチでは特定のカーネルがビジーであることが認識され、別のカーネルが（最大 5 まで）インスタンス化されます。新規カーネルが jde.ini ファイル内で定義されていれば、XML ディスパッチはそのカーネルの定義（XAPI など）を認識できます。新規カーネルを追加する際に JDENET コードを変更する必要はありません。

XML ディスパッチは、ERP がサポートしている全プラットフォーム上で使用可能です。

XML ディスパッチ・プロセス

XML ディスパッチは、トランスポート・ドライバまたは他の jdenet_n から、XML ドキュメント形式の標準 JDENET メッセージを受信します。トランスポート・ドライバと XML ディスパッチ間の通信は、JDENET API を使用するローカルのプロセス間通信(IPC)です。XML ディスパッチと XTS 間、および XML ディスパッチと XML カーネル間では、JDENET API を使用する IPC またはリモート・ネットワークによる通信が可能です。

XML ディスパッチは XML ドキュメントを解析してから、処理のために該当する ERP カーネルに送信します。

XML ディスパッチ・リコグナイザ

XML ディスパッチは、リコグナイザを使用して着信および発信 XML ドキュメントの処理方法を決定します。XML ディスパッチが着信 XML ドキュメントを ERP のネイティブ XML フォーマットとして認識すると、その XML ドキュメントが解析され、該当するカーネルに送信されます。発信ドキュメントの場合、リコグナイザは ERP の XML ドキュメントを ERP のネイティブ XML フォーマットのまま使用できるか、または変換が必要であるかを決定します。

さまざまな XML 構文を認識できるように、XML ディスパッチに複数のリコグナイザを追加できます。XML ディスパッチでは、次のタイプが認識されます。

- jdeRequest
- jdeResponse

- jdeWorkflow

ドキュメントの識別時に DocIsRecognized 例外が発生すると、XML ディスパッチ・リコグナイザはそれ以上の解析を停止します。

他のタイプの XML ドキュメントを認識できるリコグナイザを記述できます。タイプ指定は jde.ini ファイル内で構成します。

XML DispatchTransports

XML ディスパッチの一部としてトランスポートを記述できます。トランスポートは、MQSeries、MSMQ、HTTP、TCP/IP などのメカニズムを使用して外部システムと通信します。トランスポート・プロセスは、XML ディスパッチと同じマシン上で実行する必要があります。ERP と通信するカスタム・トランスポートを開発するには、次の API を使用します。

- jdeTransportInit
- jdeTransportMessagePut
- jdeTransportMessageGet
- jdeTransportDoIExit

トランスポート API はポーリング・モデルを想定しています。これは、メッセージを送受信するための呼出しにタイムアウトが発生しないことを意味します。トランスポート API については、オンラインのリファレンス・ガイドを参照してください。

XML ディスパッチの jde.ini ファイルの構成

XML ディスパッチ・カーネルは、jde.ini ファイル内で定義する必要があります。XML ディスパッチの設定は、次のとおりです。

```
[JDENET_KERNEL_DEF22]
    krnlName=XML DISPATCH KERNEL
    dispatchDLLName=xmldispatch.dll
    dispatchDLLFunction=_XMLDispatch@28
    maxNumberOfProcesses=1
    numberOfAutoStartProcesses=1
```

上記の設定は Windows 2000 と Windows NT の場合です。他のプラットフォームを使用している場合は、dispatchDLLName と dispatchDLLFunction に次の設定を使用してください。

プラットフォーム	dispatchDLLName	dispatchDLLFunction
AS/400	XMLDSPATCH	JDEK_XMLDispatch
HP9000	libxmldispatch.sl	JDEK_XMLDispatch
SUN または RS6000	libxmldispatch.so	JDEK_XMLDispatch

XML ディスパッチは、jde.ini ファイルの[XMLLookupInfo]セクションの設定を使用して、XML ドキュメントを対応する XML カーネルにルーティングします。XML 要求と XML カーネルで構成されるペアをマップするために、3 つのキーワード（XMLRequestN、XMLKernelMessageRangeN、および XMLKernelHostN）が使用されます。次の表に、[XMLLookupInfo]セクションの設定を示します。

設定	目的
XMLRequestTypeN=	処理するメッセージのタイプを識別します。
XMLKernelMessageRangeN=	カーネル・メッセージの範囲を識別するハードコード化された番号。
XMLKernelHostNameN=	ホスト名。
XMLKernelPortN=	値は 0 または 1 です。ローカル・ホストを指定するには"0"、リモート・ホストを指定するには"1"を入力します。
XMLKernelRplyN=	値は 0 または 1 で、デフォルトは 1 です。値 0 は、返信不要であることを示します。値 1 は、入力者に返信する必要があることを示します。 **注:** list、callmethod、および trans の場合、XMLKernelRplyN 設定は不要です。返信設定はデフォルトで 1 に設定されます。 XMLService は応答を送信せず、XMLKernelReplyN の設定はゼロ(0)にする必要があります。

N は 1 以上の値で、このセクションでは各キーのグループを複数指定できます。

[XMLLookupInfo]セクションには、次のように 6 つのグループを指定する必要があります。

```
[XMLLookupInfo]
    XMLRequestType1=list
    XMLKernelMessageRange=5257
    XMLKernelHostName1=local
    XMLKernelPort1=0

XMLRequestType2=callmethod
    XMLKernelMessageRange2=920
    XMLKernelHostName2=local
    XMLKernelPort2=0

XMLRequestType3=trans
    XMLKernelMessageRange3=5001
    XMLKernelHostName3=local
    XMLKernelPort3=0

XMLRequestType4=JDEMSGWFINTEROP
    XMLKernelMessageRange4=4003
    XMLKernelHostName4=local
    XMLKernelPort4=0
    XMLKernelReply4=0

XMLRequestType5=xapicallmethod
    XMLKernelMessageRange5=14251
    XMLKernelHostName5=local
```

```
        XMLKernelPort5=0
        XMLKernelReply5=0

    XMLRequestType6=realTimeEvent
        XMLKernelMessageRange6=14251
        XMLKernelHostName6=local
        XMLKernelPort6=0
        XMLKernelReply6=0
```

XML ディスパッチのエラー処理

XML ディスパッチでは、3 タイプのエラーが処理されます。次の表に、各エラーと XML ディスパッチでの処理方法を示します。

XML ディスパッチ、XTS、および XL カーネル・プロセスのデータ交換中にエラーが発生した場合。たとえば、通信が切断された場合。	XML ディスパッチでは、エラーを記述した XML ドキュメント形式のエラー・レポートが生成されます。
パーサーまたは XTS で XML ドキュメントの処理中にエラーが発生した場合。たとえば、構文エラー、無効な要求など。	XML ディスパッチでは、パーサーまたは XTS により生成されるエラー・メッセージに基づいて、エラー・レポートが生成されます。
XML カーネルで XML ドキュメントの処理中にエラーが発生した場合。たとえば、無効なユーザー名、トランザクションのロールバックなど。	XML ディスパッチでは、XML カーネルで生成されたエラー・レポートが、必要に応じて XTS を使用して変換されます。

XML ディスパッチは、生成したエラー・レポートを対応するトランスポート・プロセスに送信します。

XML CallObject

XML CallObject は、ERP カーネル・プロセスとして実行される XML ベースのインタオペラビリティ機能です。XML CallObject をメッセージング・アダプタと共に使用することもできます。次のリストに、XML CallObject の機能の一部を示します。

- XML ドキュメントを使用して ERP に対してビジネス関数の呼出しを実行できます。
- ビジネス関数テンプレートが用意されており、独自のテンプレートも作成できます。
- 1 つの XML ドキュメントで複数のビジネス関数を呼び出すことができます。
- COM や Java API を使用するよりも単純な方法で ERP とのインターフェイスを提供します。

XML CallObject テンプレート

XML CallObject にはブランクのテンプレートが用意されており、必要な情報を指定して特定のビジネス関数に関する CallObject 要求を実行できます。また、独自のカスタム XML ドキュメントを作成するオプションもあります。

特定のビジネス関数用の XML テンプレートを要求するには、callMethod 要求タイプの XML ドキュメントを作成します。CallObject テンプレート要求を発行すると、応答としてテンプレートが戻されます。このテンプレートにはすべての関数パラメータ情報が含まれていますが、データ値は入力されていま

せん。ユーザー、パスワード、およびセッション属性の値は、応答をキャッシュに入れて後で使用できるようにブランクになっています。

CallObject テンプレート要求は、jdeRequest が jdeResponse を戻すという規則の例外です。データの代わりにテンプレートが戻されるため、それを使用して別の CallMethod 要求を発行します。CallObject テンプレートを要求する場合、XML ドキュメントで発行できるのは、そのテンプレートに対する要求のみです。XML ドキュメントには、ビジネス関数を組み込む必要があります。

次の例に、CallObject テンプレートに対する要求を示します。

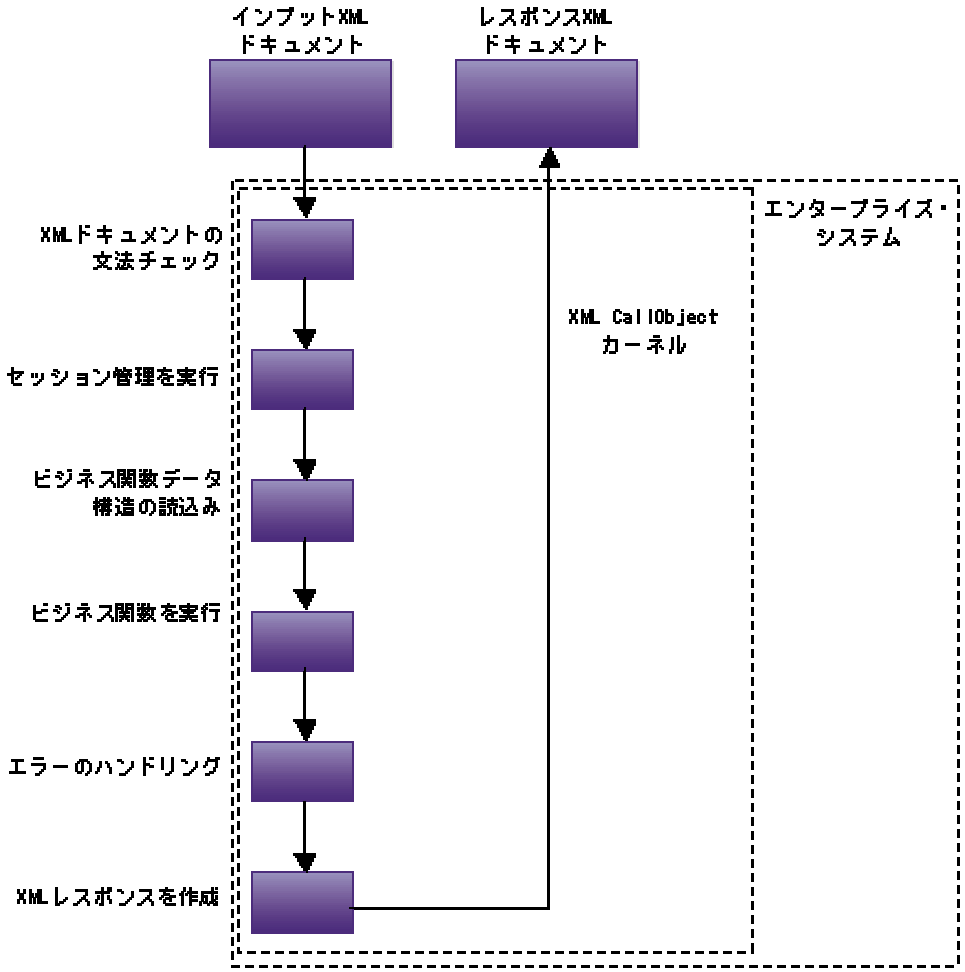
```
<?xml version='1.0' ?>
<jdeRequest type='callmethod' user='steve' pwd='xyz' en
vironment='prod' session=''>
<callMethodTemplate name='myfunc' app='P42101'/>
</jdeRequest>
```

次の例に、CallObject テンプレート要求に対する応答を示します。この応答に適切な情報を入れ、要求として送り返すことができます。

```
<?xml version='1.0' ?>
<jdeRequest type='callmethod' user='' pwd='' environ
ment='prod' session=''>
<callMethod name='myfunc' app='P42101'>
<params>
<param name='CostCtr'></param>
<param name='ExpDate'></param>
<param name='Quantity'></param>
</params>
</callMethod>
</jdeRequest>
```

XML CallObject プロセス

次の図に、XML CallObject プロセスを示します。



- エンタープライズ・サーバーが XML ドキュメントを受信します。
- XML CallObject が、XML ドキュメントを解析してメッセージを処理します。
- セッション・マネージャがユーザーとパスワードの妥当性を検査します。
- 要求されたビジネス関数は、すべての呼出しが処理されるまで、それぞれ個別に呼び出されるか、要求されたトランザクション境界内で呼び出されます。
- 出力データとエラー・メッセージが入力 XML ドキュメントのデータとマージされ、新しい応答ドキュメントが作成され、入力者に送信されます。

XML CallObject の要素

XML ドキュメントの先頭には、次の要素を指定する必要があります。

- jdeRequest タイプ
- セッションの確立
- セッションの満期

XML ドキュメントの終わりには、次の要素を指定する必要があります。

- セッションの終了

XML CallObject ドキュメントには、必要に応じて次の要素も指定できます。

- オブジェクトの呼出し
- OnError の処理
- オブジェクト呼出しエラー処理
- エラー・テキスト
- ドキュメントごとの複数要求
- ID/IDREF サポート
- NULL 値のリターン

オブジェクトの呼出し

タグを使用して、サーバー上でビジネス関数を呼び出します。

ここでは、callObject の使用例を示して使用方法を説明します。

```
<?xml version='1.0' ?>
<jdeRequest type='callmethod' user='steve' pwd='xyz' environ
ment='prod'>
<callMethod name='myfunc' app='P42101'>
<params>
<param name='CostCtr'>      1001</param>
<param name='ExpDate'>1999/10/31</param>
<param name='Quantity'>12</param>
</params>
</callMethod>
</jdeRequest>
```

callMethod 要素は、呼び出す関数と、それが呼び出されるコンテキストを詳しく説明します。name 属性は呼び出すビジネス関数を指定し、app 属性は誰が呼び出しているのかをビジネス関数に知らせます。

params および param 要素は、ビジネス関数のデータ構造体を定義します。各 param 要素はデータ構造体メンバをそれぞれ 1 つずつ説明します。コール元が必要になるのは、name 属性を指定するときだけです。

入力データ構造体メンバに対して param 要素値を指定しない場合、値は NULL またはゼロと同じように扱われます。

OnError の処理

callMethod 要求に onError 要素を追加して、エラーが発生した場合に特定の処理を実行できます。onError タグで、後続の要求をすべてスキップするかどうかを指定する abort 属性を指定できます。有効な値は yes または no です。global の onError タグは、jdeRequest タグの子として指定できます。このタグは、エラーが発生したときに、abort='yes'が設定されて実行される onError タグがない場合に実行されます。'global'の onError タグは、ドキュメントにおいて、OnError タグが実行されにくい要求を処理することになります。

```
<?xml version='1.0' ?>
<jdeRequest type='callmethod' user='steve' pwd='xyz' environ
ment='prod' session=''>
<callMethod name='myfunc' app='P42101' trans='t1' runOnEr
ror='yes'>
<params>
<param name='CostCtr'>      1001</param>
</params>
<onError abort='no'>
<endTransaction trans='t1' action='rollback'/>
</onError>
</callMethod>
</jdeRequest>
```

オブジェクト呼出しエラー処理

オブジェクト呼出しのシステム・エラーは、returnCode 要素で報告されます。code 属性で数値コードが戻され、対応するテキストが returnCode 要素の子テキスト・ノードとして戻されます。code 属性に対しては、標準 jdeCallObject リターン・コードが使用されます。

```
<?xml version='1.0' ?>
<jdeResponse type='callmethod' user='steve' pwd='xyz' environ
ment='prod' session=''>
<callMethod name='myfunc' app='P42101' trans='t1'>
<params>
<param name='CostCtr'>      1001</param>
</params>
<returnCode code='0'>Success</returnCode>
</callMethod>
</jdeResponse>
```

エラー・テキスト

エラー・テキストを使用して、ビジネス関数のエラーを処理できます。エラー・テキストを使用すると、ビジネス関数のエラー・メッセージが errors 要素内で戻されます。errors 要素内には、エラー・コードの code 属性およびエラー・テキストのある子テキスト・ノードを含め、ゼロ個以上のエラー要素があります。name 属性は、エラーで参照されている param 要素の記述です。

```
<?xml version='1.0' ?>
<jdeResponse type='callmethod' user='steve' pwd='xyz' environ
ment='prod' session=''>
<callMethod name='myfunc' app='P42101' trans='t1'>
<params>
<param name='CostCtr'>      1001</param>
</params>
<returnCode code='2'>Errors</returnCode>
<errors>
   <error code='192' name='CostCtr'>Cost Center not valid</er
ror>
</errors>
</callMethod>
</jdeResponse>
```

ドキュメントごとの複数要求

複数の要求を XML ドキュメントに組み込むことができます。デフォルトでは、前の要求でエラーが発生すると、要求は実行されません。エラーが発生しても要求を実行させるには、要求 runOnError 属性で値 yes を指定してデフォルトの動作を上書きします。

```
<?xml version='1.0' ?>
<jdeRequest type='callmethod' user='steve' pwd='xyz' environ
ment='prod' session=''>
<callMethod name='myfunc' app='P42101' trans='t1' runOnEr
ror='yes'>
<params>
<param name='CostCtr'>      1001</param>
</params>
</callMethod>
</jdeRequest>
```

ID/IDREF サポート

ID type 属性は、XML ドキュメント内の要素をストリング値で独自に識別します。IDREF 属性を使うと、指定された要素を他の要素が参照できます。IDREF 属性は、それが参照する ID を定義するまでは、ドキュメントで使用しないでください。

param 要素は、callMethod 要求からの出力値を保存し、後で IDREF 属性によって別の param 要素で参照できるように、ID 属性を指定できます。param 要素に IDREF 属性が含まれている場合は、指

定したパラメータの値を param 要素の入力値として使用します。たとえば、参照されるパラメータの出力値を XML の値の代わりに使います。

```
<?xml version='1.0' ?>
<jdeRequest type='callmethod' user='steve' pwd='xyz' environ
ment='prod' session=''>
<callMethod name='myfunc' app='P42101' trans='t1' runOnEr
ror='yes'>
<params>
<param name='CostCtr'>      1001</param>
<param name='Company1' id='c1'></param>
<param name='Company2' id='c2'></param>
</params>
</callMethod>
<callMethod name='myfunc2' app='P42101' trans='t1' runOnEr
ror='yes'>
<params>
<param name='Company1' idref='c1'></param>
</params>
<returnParams><param idref='c2'/></returnParams>
</callMethod>
</jdeRequest>
```

returnParams という特殊要求タグを指定できます。これには、1 つまたは複数の param 要素を組み込むことができます。param 要素に IDREF 属性が含まれている場合は、参照された値が応答にコピーされます。

NULL 値のリターン

デフォルトでは、要求ドキュメントでパラメータを指定しない場合、その値が非ブランクまたは非ゼロでない限り、要求ドキュメントでパラメータは戻されません。この動作は、callMethod 要素の returnNullData 属性に yes を指定することで変更できます。

```
<?xml version='1.0' ?>
<jdeRequest type='callmethod' user='' pwd='' environment='prod'
session=''>
<callMethod name='myfunc' app='P42101' returnNullData='yes'>
<params>
<param name='CostCtr'></param>
<param name='ExpDate'></param>
<param name='Quantity'></param>
</params>
</callMethod>
</jdeRequest>
```

XML CallObject の jde.ini ファイルの構成

XML callobject カーネルの jde.ini 設定は次のようになります。

```
[JDENET_KERNEL_DEF6]
    krnlName=CALL OBJECT KERNEL
    dispatchDLLName=XMLCallObj.dll
    dispatchDLLFunction=_XMLCallObjectDispatch@28
    maxNumberOfProcesses=1
    numberOfAutoStartProcesses=1
```

他のプラットフォームのさまざまな.dll 拡張子は、次の表を参照してください。

プラットフォーム	dispatchDLLName	dispatchDLLFunction
AS/400	XMLCALLOBJ	XMLCallObjectDispatch
HP9000	libxmlcallobj.sl	XMLCallObjectDispatch
SUN または RS6000	libxmlcallobj.so	XMLCallObjectDispatch

例 : CallObject 要求

次のサンプル・コードに、CallObject 要求を示します。

```
<?xml version="1.0" encoding="utf-8" ?>
<jdeRequest pwd="JDE" type="callmethod" user="JDE" session="" environment="M7333NIS2"
sessionidle="1800">
  <callMethod app="XMLTest" name="AddressBookMasterMBF">
    <params>
      <param name="cActionCode">A</param>
      <param name="cUpdateMasterFile">1</param>
      <param name="mnAddressBookNumber" idref="ABNumber" />
      <param name="szSearchType">C</param>
      <param name="szAlphaName">bobs</param>
      <param name="szMailingName">Bob's Shrimp boats</param>
      <param name="szAddressLine1">One Technology Way</param>
      <param name="szPostalCode">80237</param>
      <param name="szCity">Denver</param>
      <param name="szCounty">Denver</param>
      <param name="szState">CO</param>
      <param name="szCountry">US</param>
      <param name="cPayablesYNM">N</param>
      <param name="cReceivablesYN">Y</param>
      <param name="cEmployeeYN">N</param>
      <param name="cUserCode">N</param>
      <param name="cARAPNettingY">N</param>
      <param name="jdDateEffective">01/23/2001</param>
      <param name="szProgramId">EP01012</param>
      <param name="mnAddNumParentOriginal">0</param>
      <param name="szVersionconsolidated" idref="Version" />
      <param name="szCountryForPayroll">US</param>
    </params>
```

```
  </callMethod>
</jdeRequest>
```

例：CallObject 応答

次のサンプル・コードに、CallObject 応答を示します。

```
<?xml version="1.0" encoding="UTF-8" ?>
<jdeResponse pwd="JDE" role="*ALL" type="callmethod" user="JDE" session="2360.1049473980.6"
environment="PDEVNIS2" sessionidle="1800">
<callMethod app="XMLTest" name="AddressBookMasterMBF">
 <returnCode code="0" />
  <params>
   <param name="cActionCode">A</param>
   <param name="cUpdateMasterFile">1</param>
   <param name="mnAddressBookNumber">57322</param>
   <param name="szSearchType">C</param>
   <param name="szAlphaName">bobs</param>
   <param name="szMailingName">Bob's Shrimp boats</param>
   <param name="szBusinessUnit">1</param>
   <param name="szAddressLine1">One Technology Way</param>
   <param name="szPostalCode">80237</param>
   <param name="szCity">Denver</param>
   <param name="szState">CO</param>
   <param name="szCountry">US</param>
   <param name="cPayablesYNM">N</param>
   <param name="cReceivablesYN">Y</param>
   <param name="cEmployeeYN">N</param>
   <param name="cUserCode">N</param>
   <param name="cARAPNettingY">N</param>
   <param name="cAddressType3YN">N</param>
   <param name="cAddressType4YN">N</param>
   <param name="cAddressType5YN">N</param>
   <param name="jdDateEffective" />
   <param name="szProgramId">EP01012</param>
   <param name="szVersionconsolidated">ZJDE0001</param>
   <param name="cEdiSuccessfullyProcess">0</param>
   <param name="szCountryForPayroll">US</param>
  </params>
 </callMethod>
</jdeResponse>
```

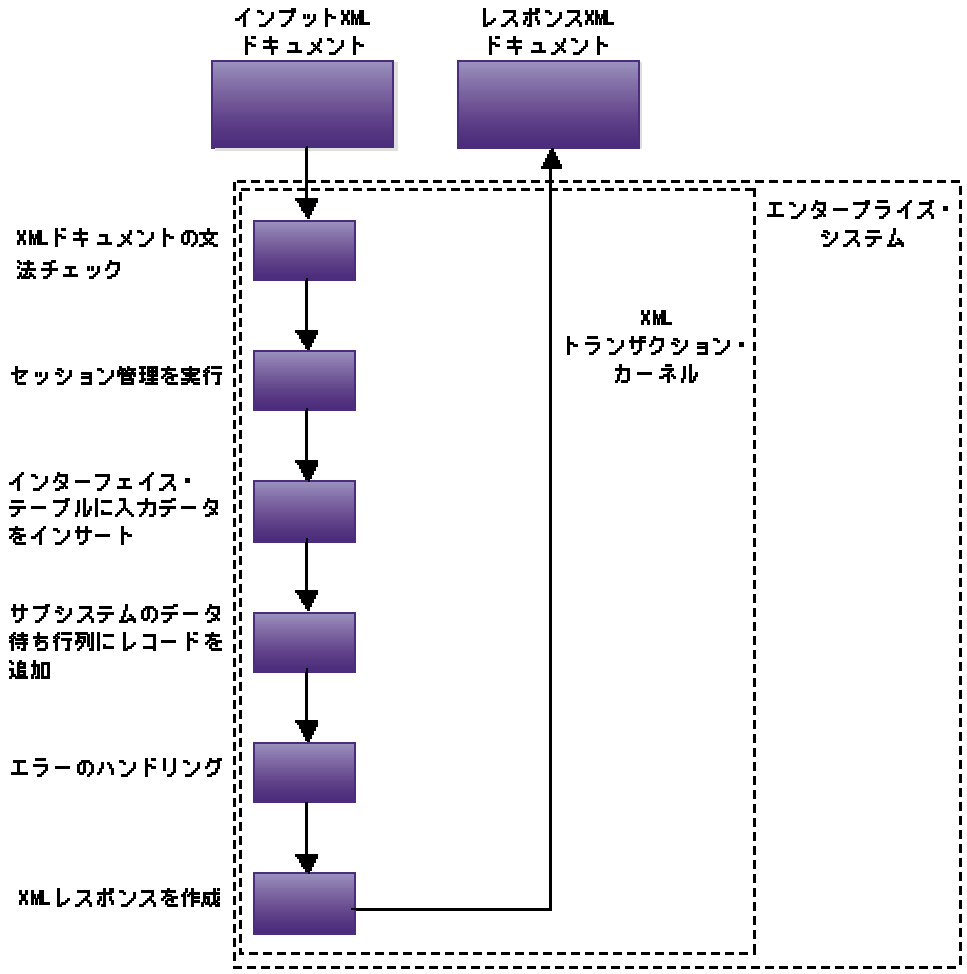
XML トランザクション

XML トランザクションは、ERP カーネル・プロセスとして実行される XML ベースのインタオペラビリティ機能です。XML トランザクションをメッセージング・アダプタと共に使用することもできます。XML トランザクションはインターフェイス・テーブル（Z テーブル）と対話して、ERP データベースを更新するか、ERP データを取得します。1 つの XML ドキュメントに ERP の更新と ERP からのデータ取得の両方を組み込んで作成できます。

XML トランザクションの更新プロセス

ERP にデータを挿入するには、フォーマット済みの XML ドキュメントを使用します。この XML ドキュメントには、JDEPOIN など、事前定義済みのトランザクション・タイプが含まれています。1 つまたは複数の ERP インターフェイス・テーブルが識別され、更新対象の全データ（データ・タイプと実際のデータ値）がリストされます。

次の図に、XML トランザクションの更新プロセスを示します。

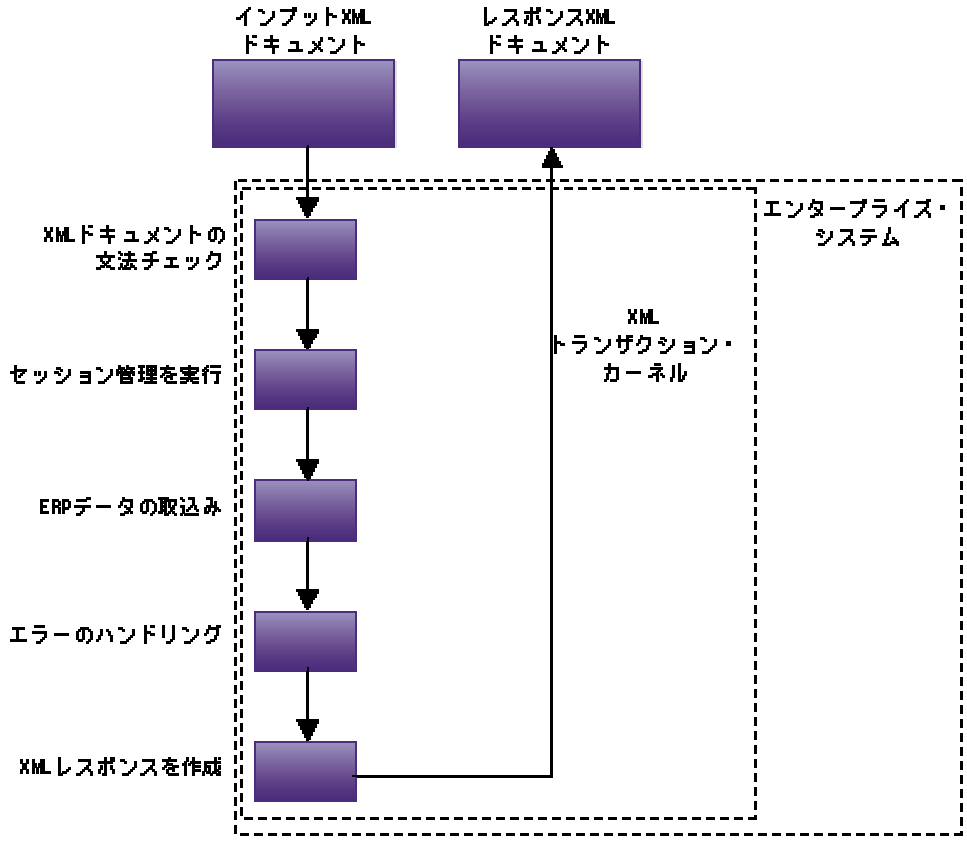


- XML ドキュメント形式の要求には、事前定義済みのトランザクション・タイプに関するデータのリストが含まれています。
- XML トランザクションは XML 受信ドキュメントを解析し、そのデータを ERP 受信インターフェイス・テーブルに挿入します。
- XML トランザクションはサブシステムのデータ待ち行列レコードを追加して、そのレコードを処理するように ERP サブシステムに通知します。
- インターフェイス・テーブルへの挿入とサブシステム・データ待ち行列の追加に成功したかどうかを示す応答が、要求側に送信されます。

XML トランザクションのデータ要求プロセス

ERP からのデータを要求するには、フォーマット済みの XML ドキュメントを使用します。XML ドキュメントには、JDESOUT などのトランザクション・タイプと、ERP インターフェイス・テーブルからの取得データを識別するインデックスが含まれています。特定のデータを取得するには、テンプレートを提供します。

次の図に、XML トランザクションのデータ要求および応答プロセスを示します。



- XML ドキュメント形式の要求には、トランザクション・タイプと要求するデータのインデックスが含まれています。
- XML トランザクションは XML 受信ドキュメントを解析し、トランザクション・タイプとインデックスを取得します。
- XML トランザクションは ERP からデータを取得して、ERP インターフェイス・テーブルに挿入します。
- XML トランザクションは XML ドキュメント形式で応答を作成します。応答は、トランザクション・タイプおよびインデックスが一致するインターフェイス・テーブルのデータ・レコードで構成されます。エラーが発生した場合は、応答にエラー・メッセージも含まれます。

XML トランザクションの jde.ini ファイルの設定

XML トランザクション・カーネルの jde.ini 設定は次のようになります。

```
[JDENET_KERNEL_DEF15]
    krnlName=XML TRANSACTION KERNEL
    dispatchDLLName=XMLTransactions.dll
    dispatchDLLFunction=_XMLTransactionDispatch@28
    maxNumberOfProcesses=1
    numberOfAutoStartProcesses=1
```

他のプラットフォームのさまざまな.dll 拡張子は、次の表を参照してください。

プラットフォーム	dispatchDLLName	dispatchDLLFunction
AS/400	XMLTRANS	XMLTransactionDispatch
HP9000	libxmltransactions.sl	XMLTransactionDispatch
SUN または RS6000	libxmltransactions.so	XMLTransactionDispatch

例:送信オーダー状況 XML 要求と応答フォーマット

XML トランザクション・データ要求は、送信関数によって作成され、XML トランザクション API に送信されます。ここでは、受注オーダー要求および応答のサンプル・コードを示します。

次のサンプル・コードは、XML トランザクション要求を示しています。

```
"This format will return all columns for the sales order header (F4201Z1) and detail lines (F4211Z1). "

<?xml version='1.0' ?>
<jdeRequest type='trans' user='user' pwd='password' environment='environment' session='
` sessionidle='300'>
<transaction action='transactionInfo' type='JDESOOUT'>
<key>
<column name='EdiUserId'>value</column>
<column name='EdiBatchNumber'>value</column>
<column name='EdiTransactNumber'>value</column>
</key>
</transaction>
</jdeRequest>
```

次のサンプル・コードは、XML トランザクション応答を示しています。

```
<?xml version='1.0' encoding='utf-8' ?>
<jdeResponse type='trans' user='user' session='session1' environment='env'>
    <transaction type='JDESOOUT' action='transactionInfo'>
        <returnCode code='0'>XML Request OK</returnCode>
        <key>
            <column name='EdiUserId'></column>
            <column name='EdiBatchNumber'></column>
```

```
                    <column name='EdiTransactNumber'></column>
            </key>
            <table name='F4201Z1' type='header'>
                    <column name='EdiUserId'></column>
                    <column name='EdiBatchNumber'></column>

            </table>
            <table name='F4211Z1' type='detail'>
                    <column name='EdiUserId'></column>
                    <column name='EdiBatchNumber'></column>

            </table>
            <table name='F49211Z1' type='additionalHeader'>
                    <WARNING>No record found</WARNING>
            </table>
    </transaction>
</jdeResponse>
```

XML リスト

XML リストは、ERP カーネル・プロセスとして実行される XML ベースのインタオペラビリティ機能です。XML リストには、List/GetNext 機能が用意されており、ERP からレコードのリストを収集できます。XML リストは、ERP テーブル変換(TC)エンジンにビルドされます。XML ドキュメントを要求として受け取り、要求されたデータを XML ドキュメントに入れて返します。データは、テーブル、ビジネス・ビュー、またはテーブル変換によるデータとして表示できます。テーブル変換のデータを使用すると、複数のテーブルを使用できます。XML ドキュメントを送信することで、リストのメタデータの取得、リストの作成、リストのデータのチャンクの取得、またはリストの削除を実行できます。JDENet またはサードパーティ・ソフトウェアを使用して要求を送信し、次の操作のいずれかを実行できます。

- リストの作成
- テンプレートの取得
- グループの取得
- リストの削除

XML リストには、通常および特殊な List/GetNext API があります。通常の List/GetNext API は、単一テーブルでのデータの選択など、単純な取得を実行します。特殊な List/GetNext API では、イベント・ルールなどの追加機能が使用されます。特殊な List/GetNext BPAPI には、それぞれ独自にテーブル変換が設計されている必要があります。データ選択とデータ順序設定は、実行時に XML 要求で定義できます。

XML リストにはリスト検索エンジンが用意されており、XML データ・ファイルをシステム・リポジトリに作成し、データを小さい単位で取得できます。

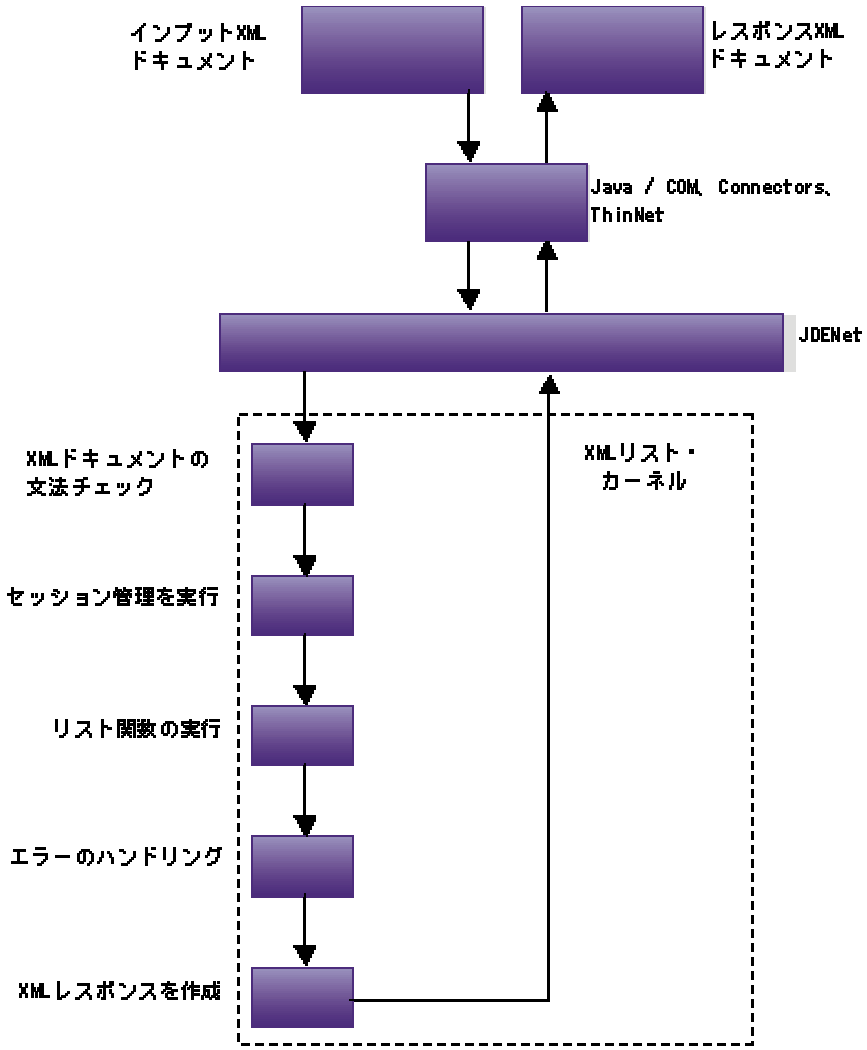
リスト検索エンジンのテーブル変換ラッパー

リスト検索エンジンは、XML リポジトリ・ファイルへのアクセスを提供および管理する、最適化されたデータベース・エンジンです。各 XML リスト・リポジトリ・ファイルは、インデックス・ファイルとデータ・ファイルの対になります。それぞれのファイルの拡張子は、*.idb と*.ddb です。.idb ファイルは、データ・ファイルに生成されたインデックスを保持します。また、.ddb ファイルは、テーブル変換エンジンに

よって生成されたデータを保持します。TCWrapper は、TCPEngine およびリスト検索エンジンからのリスト検索およびリスト処理 API を総合するシステム・モジュールで、XML リストのデータに対して同一のアクセスを可能にします。

XML リスト・プロセス

次の図に、通常および特殊な XML リスト要求に対する XML リスト・プロセスを示します。



- JDENet が XML ドキュメントを受信します。
- JDENet が XML ドキュメントを XML リスト・カーネルに渡します。
    - CreateList または GetTemplate に関する要求の場合、XML リストはセッションを作成します。

- 通常の要求の場合、XML リストはデータを取得して、要求側に送信する応答メッセージを作成します。
- 特殊な要求の場合、XML リスト・カーネルは要求を次の該当する API に渡します。
  - GetTemplate
  - CreateList
  - GetGroup
  - DeleteList

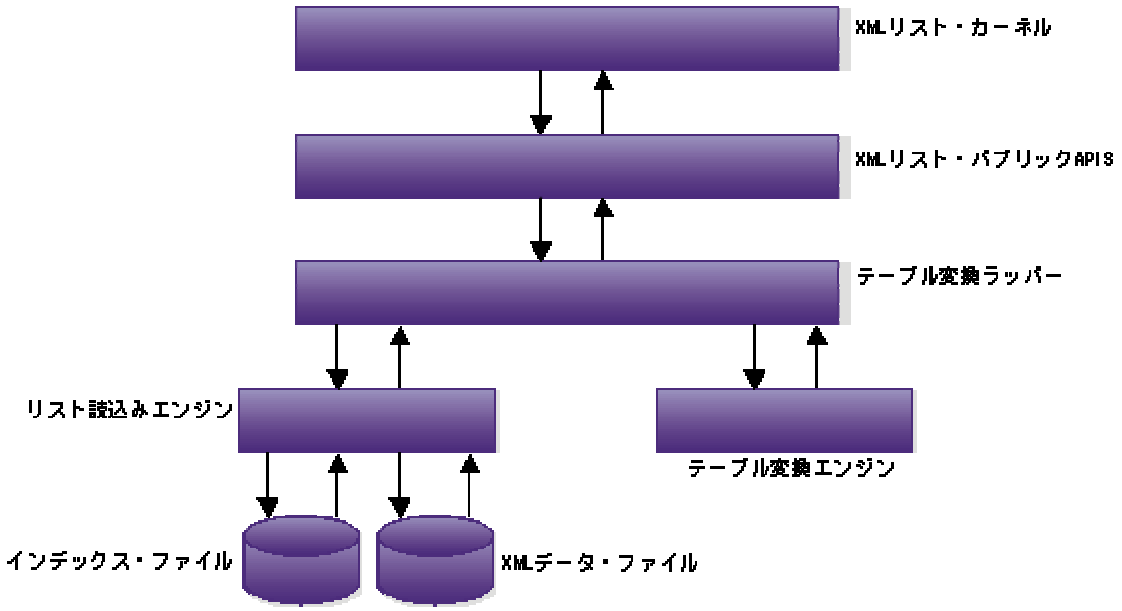
- テーブル変換ラッパーは、特殊な要求の結果として取得されたデータを処理します。テーブル変換ラッパーは、テーブル変換エンジンとリスト検索エンジンからリスト検索およびリスト処理 API を総合し、データに対して同一のアクセスを可能にします。

XML リスト要求

XML リストを使用して次のいずれかの要求を発行できます。

GetTemplate	CreateList 要求にデータ選択とデータ順序設定を追加できるように、リストのメタデータ情報の取得要求を送信します。
CreateList	データ選択および順序設定と共に TC/Table name を指定して要求を送信します。応答は XML ドキュメントで、リポジトリ内で作成されたリストに関連するハンドルとサイズが含まれています。
GetGroup	以前の CreateList 要求で生成されたリストからデータを取得するための要求を送信します。GetGroup は、ハンドル値と取得するレコードの範囲を渡します。
DeleteList	リポジトリからリストを削除するための要求を送信します。

次の図に、リスト操作に関連する各種コンポーネントを示します。



XML リスト要求を使用すると、次の処理を実行できます。

- リストの作成
- リストのデータの取得
- リストの削除
- リストのカラム情報の取得

リストの作成

次の例は、XML 要求での CreateList の使用法を示しています。TC Name/Table Name、およびデータの選択と順序設定を指定しています。XML 応答に、作成されたリストに関連付けられたハンドルを返します。

```
<?xml version="1.0"?>
<jdeRequest type="list" user="JDE" pwd="JDE" environment="PRODHP01" session="" sessionidle="">
    <ACTION TYPE="CreateList">
      <TC_NAME VALUE=""/>
      <TC_VERSION VALUE=""/>
      <FORMAT VALUE="UT"/>
        <RUNTIME_OPTIONS>
        <DATA_SELECTION>
          <CLAUSE TYPE="WHERE">
            <COLUMN NAME="" TABLE="" INSTANCE="" ALIAS=""/>
            <OPERATOR TYPE="EQ"/>
            <OPERAND>
              <COLUMN NAME="" TABLE="" INSTANCE="" ALIAS=""/>
              <LITERAL VALUE=""/>
              <LIST>
                <LITERAL VALUE=""/>
              </LIST>
              <RANGE>
                <LITERAL_FROM VALUE=""/>
                <LITERAL_TO VALUE=""/>
              </RANGE>
            </OPERAND>
          </CLAUSE>
<CLAUSE TYPE="OR">
            <COLUMN NAME="" TABLE="" INSTANCE="" ALIAS=""/>
            <OPERATOR TYPE="EQ"/>
            <OPERAND>
              <COLUMN NAME="" TABLE="" INSTANCE="" ALIAS=""/>
              <LITERAL VALUE=""/>
              <LIST>
                <LITERAL VALUE=""/>
              </LIST>
              <RANGE>
                <LITERAL_FROM VALUE=""/>
                <LITERAL_TO VALUE=""/>
              </RANGE>
            </OPERAND>
          </CLAUSE>
        </DATA_SELECTION>
```

```
        <DATA_SEQUENCING>
          <DATA SORT="ASCENDING">
            <COLUMN NAME="Product Code" TABLE="F0004" INSTANCE="" ALIAS=""/>
          </DATA>
        </DATA_SEQUENCING>
      </RUNTIME_OPTIONS>
    </ACTION>
</jdeRequest>
```

TC_NAME と TC_VERSION、または TABLE_NAME と TABLE_TYPE のいずれかを、要求に定義する必要があります。TABLE_TYPE には、次のいずれかを指定します。

- OWTABLE
- OWVIEW
- FOREIGN_TABLE

CLAUSE には、WHERE、OR、または AND を指定して、SQL ステートメントをシミュレートします。

COLUMN NAME に意味のある名前を指定して、ALIAS で定義されるテーブルの実際のカラム名をわかりやすくできます。TABLE、INSTANCE、および ALIAS の値は、GetTemplate 要求で返される XML 応答と同じにします。たとえば、カラム X がデータ選択にある場合、GetTemplate 要求によって次の情報が返されるため、<COLUMN NAME="My column" TABLE="F9999" INSTANCE="0" ALIAS="X"/>を指定します。

```
<COLUMN NAME="X" ALIAS="X" TYPE="String" LENGTH="32" TABLE="F9999"
    INSTANCE="0">
```

OPERATOR には、EQ、NE、LT、GT、LE、GE、IN、NI、BW (between)、または NB を使用します。

OPERAND ノードには、サポートされている次の要素タイプを 1 つ含めることができます。

- Column
- Literal
- List
- 範囲

次の XML ノードに、OPERAND ノードでサポートされる要素を太字で示します。これはテンプレート・フラグメントであり、サポートされる要素は 1 つのみ使用する必要があります。

```
<CLAUSE TYPE="WHERE" >
    <COLUMN NAME="UserDefinedCodes" TABLE="F0005" INSTANCE="" ALIAS="RT" />
    <OPERATOR TYPE="EQ"/>
    <OPERAND>
            <COLUMN NAME="" TABLE="" INSTANCE="" ALIAS="null"/>
            <LITERAL VALUE="P4"/>
            <RANGE>
            </RANGE>
    </OPERAND>
</CLAUSE>
```

次のサンプル XML ノードに、サポートされているさまざまな要素を使用して演算子タイプ（太字）とオペランド（太字）を示します。

オペランドが COLUMN の場合、COLUMN 要素を指定します。たとえば、次のように記述します。

```
<CLAUSE TYPE="WHERE">
    <COLUMN NAME="DRSY" TABLE="F0005" INSTANCE="0" ALIAS="SY"/>
    <OPERATOR TYPE="EQ"/>
    <OPERAND>
        <COLUMN NAME="DRRT" TABLE="F0005" INSTANCE="0" ALIAS="RT"/>
    </OPERAND>
</CLAUSE>
```

オペランドが LITERAL の場合、LITERAL 要素を指定します。

```
<CLAUSE TYPE="WHERE">
    <COLUMN NAME="DRSY" TABLE="F0005" INSTANCE="0" ALIAS="SY"/>
    <OPERATOR TYPE="EQ"/>
    <OPERAND>
        <LITERAL VALUE="08"/>
    </OPERAND>
</CLAUSE>
```

オペランドが LIST の場合、LIST 要素を指定します。LIST は、IN または NI と共に使用します。

```
<CLAUSE TYPE="WHERE">
    <COLUMN NAME="DRSY" TABLE="F0005" INSTANCE="0" ALIAS="SY"/>
    <OPERATOR TYPE="IN"/>
    <OPERAND>
        <LIST>
            <LITERAL VALUE="08"/>
            <LITERAL VALUE="09"/>
        </LIST>
    </OPERAND>
</CLAUSE>
```

オペランドが RANGE の場合、RANGE 要素を指定します。RANGE は、BW または NB と共に使用します。

```
<CLAUSE TYPE="WHERE">
    <COLUMN NAME="DRSY" TABLE="F0005" INSTANCE="0" ALIAS="SY"/>
    <OPERATOR TYPE="BW"/>
    <OPERAND>
        <RANGE>
            <LITERAL_FROM VALUE="08"/>
            <LITERAL_TO VALUE="10"/>
        </RANGE>
    </OPERAND>
</CLAUSE>
```

CreateList 要求に対する XML 応答は、次のようになります。

```
<?xml version="1.0"?>
<jdeResponse type="list" session="5665.931961929.454">
<returnCode code="0">XMLRequest OK</returnCode>
    <ACTION TYPE="CreateList">
                        <TABLE_NAME VALUE="F0005">
        <HANDLE>"1r4670001"</HANDLE>
        <SIZE>773</SIZE>
```

```
    </ACTION>
</jdeResponse>
```

HANDLE の値を公開して、GetGroup または DeleteList 要求で参照できます。

List のデータの取得

GetGroup 要求を使用すると、CreateList 要求で生成したリストからデータを取得できます。HANDLE、FROM VALUE、および TO VALUE を、要求で定義できます。

```
<?xml version="1.0"?>
<jdeRequest type="list" user="JDE" pwd="JDE" environment="PRODHPO1" >
   <ACTION TYPE="GetGroup">
      <HANDLE VALUE="lr4670001"/>
      <FROM VALUE="10"/>
      <TO VALUE="50"/>
   </ACTION>
</jdeRequest>
```

XML 応答では、指定した範囲内にあるレコードのリストが表示されます。リストで#10～#50 のレコードを、次のフォーマットで返します。デフォルトの FROM 値は、リストの最初のレコードです。また、デフォルトの TO 値は、リストの最後のレコードです。リスト全体に対する GetGroup 要求の場合、FROM および TO 値を指定する必要はありません。

```
<?xml version="1.0"?>
<jdeResponse type="list">
<returnCode code="0">XMLRequest OK</returnCode>
<ACTION TYPE="GetGroup">
<HANDLE VALUE="lr4670001"/>
      <FROM VALUE="10"/>
      <TO VALUE="50"/>
   <Format name='Standard'><Column name='X>abc</Column><Column
name='Y'>edf</Column></Format>
   00
   </ACTION>
</jdeResponse>
```

リストの削除

すべての GetGroup 要求が完了したら、リストを削除できます。

```
<?xml version="1.0"?>
<jdeRequest type="list" user="JDE" pwd="JDE" environment="PRODHPO1">
   <ACTION TYPE="DeleteList">
      <HANDLE VALUE="lr4670001"/>
   </ACTION>
</jdeRequest>
```

HANDLE で定義されたリスト結果が、ストレージから削除され、状況を含む応答がコール元に返されます。

```
<?xml version="1.0"?>
<jdeResponse type="list" >
<returnCode code="0">XMLRequest OK</returnCode>
   <ACTION TYPE="DeleteList">
<HANDLE VALUE="lr4670001"/>
      <STATUS>OK</STATUS>
   </ACTION>
</jdeResponse>
```

リストのカラム情報の取得

データ選択および順序設定を CreateList 要求に追加できるように、GetTemplate 要求を送信してリストのカラム情報を取得できます。TEMPLATE_TYPE で OUTPUT が定義されている場合、応答はテーブル変換に基づいて CreateList 要求によって生成された XML 出力のカラム情報のみとなります。通常のテーブル変換では、どちらのテンプレートも同じになります。タグが指定されていない場合、デフォルトのテンプレート・タイプは INPUT です。

```
<?xml version="1.0"?>
<jdeRequest type="list" user="JDE" pwd="JDE" environment="PRODHPO1" session=""
sessionidle="">
   <ACTION TYPE="GetTemplate">
      <TABLE_NAME VALUE="F0004"/>
      <TABLE_TYPE VALUE="OWTABLE"/>
      <TEMPLATE_TYPE VALUE="OUTPUT"/>
   </ACTION>
</jdeRequest>
```

入力テンプレートの応答により、すべてのカラムのエイリアス名、タイプ、およびデータ・タイプの長さがリスト表示されます。ただし、データ長は文字列タイプでしか意味を持ちません。

```
<?xml version="1.0"?>
<jdeResponse type="list" session="5665.931961929.454">
<returnCode code="0">XMLRequest OK</returnCode>
   <ACTION TYPE="GeTemplate">
            <TABLE_NAME VALUE="F0004"/>
      <TABLE_TYPE VALUE="OWTABLE"/>
      <TEMPLATE_TYPE VALUE="INPUT"/>
      <COLUMN Name="Address" Alias="X" TYPE="String" LENGTH="32"
TABLE="F9999" INSTANCE="0">
   </ACTION>
</jdeResponse>
```

リスト検索エンジンの jde.ini ファイルの設定

リスト検索エンジンは、事前定義済みのフォルダをシステム・ディレクトリとして使用し、リポジトリ・ファイルの保持と管理を行います。このシステム・ディレクトリは、jde.ini ファイルで次のように構成します。

[LREngine]

System=C:¥output

Repository_Size=20（空きディスク容量のうち XML リスト・リポジトリに割り当てるパーセンテージ）

Disk_Monitor=Yes（ディスク上の空き領域をモニタリング）

注:

AS400 では、エンジンは IFS ファイル・システムにします。したがって、対応する system サブセクションを設定する必要があります。

jde.ini ファイルの[SECURITY]セクションも構成する必要があります。DefaultEnvironment、Password、および User 設定を入力して、エンジンがデフォルトのユーザーを確認し、デフォルトの環境を初期化できるようにします。

XML リストの jde.ini ファイルの構成

XML リスト・カーネルの jde.ini 設定は次のようになります。

```
[JDENET_KERNEL_DEF16]
    krnlName=XML LIST
    dispatchDLLName=xmllist.dll
    dispatchDLLFunction=_XMLListDispatch@28
    maxNumberOfProcesses=3
    beginningMsgTypeRange=5257
    endingMsgTypeRange=5512
    newProcessThresholdRequest=0
    numberOfAutoStartProcesses=1
```

他のプラットフォームのさまざまな.dll 拡張子は、次の表を参照してください。

プラットフォーム	dispatchDLLName	dispatchDLLFunction
AS/400	XMLLIST	XMLListDispatch
HP9000	libxmllist.sl	XMLListDispatch
SUN または RS6000	libxmllist.so	XMLListDispatch

トラブルシューティング : XML カーネル

1 つまたは複数の XML カーネルが正常に動作しない場合は、次のトラブルシューティング・ガイドラインに従って、システムが正しくセットアップされているかどうかを確認してください。

- サーバーの jde.ini ファイル内でカーネル定義をチェックします。実行中のプラットフォームに合ったライブラリ名が設定されているかどうかと、エントリ関数名もチェックしてください。
- カーネルの起動が許可されているかどうかをチェックします。サーバーの jde.ini ファイル内で、カーネルの maxNumberOfProcesses 値と numberOfAutoStartProcesses 値をチェックしてください。カーネルを自動起動する必要はありません。特定のカーネルを処理するには、許容プロセス数を 1 以上に設定する必要があります。
- 特定のカーネル・タイプに対する同時要求数が大きい場合は、そのカーネルの許容プロセス数を増やしてください。これにより、要求の往復時間が短縮されるのみでなく、「Query Full」エラーもなくなります。

- XMLList カーネルを使用する場合は、サーバーの jde.ini ファイルの LREngine セクションが正しく設定されているかどうかと、指定したパスが存在するかどうかをチェックします。ERP ユーザーがこの位置への書込み許可を持っているかどうかもチェックしてください。
- XML ドキュメントが的確な XML ドキュメントであるかどうかをチェックします。そのためには、任意の XML エディタを使用するか、Microsoft Internet Explorer でドキュメントを開き、エラーの有無を調べます。
- 入力 XML ドキュメントのルートが jdeRequest であるかどうかをチェックします。すべての入力 XML ドキュメントには、ルート要素として jdeRequest が必要です。
- XML ドキュメントで有効なユーザーID、パスワード、および環境が指定されているかどうかをチェックします。
- XML ドキュメントの要求タイプが正しいかどうかをチェックします。XMLCallObject、XMLList、および XMLTransaction の各カーネルで許可される要求タイプは、それぞれ callmethod、list、および trans です。

Z トランザクション

Z トランザクションとは、ERP データベースの更新用に ERP インターフェイス・テーブル（Z テーブル）内で適切にフォーマットされている非 ERP 情報です。インターフェイス・テーブルとは、ERP アプリケーション・テーブルをミラー化するためのワークテーブルです。一部のアプリケーション・トランザクションの場合は、事前定義済みのインターフェイス・テーブルが用意されています。また、J.D. Edwards 標準に従ったフォーマットで独自のインターフェイス・テーブルを作成することもできます。

Z トランザクションは一度に 1 トランザクションずつ ERP に処理できます。この方法はバッチ・オブ・ワンと呼ばれます。また、多数のトランザクションをインターフェイス・テーブルに入れてから、一度にすべてのトランザクションを処理する方法もあります。この方法はバッチ処理と呼ばれます。

Z トランザクションを次のモデル・タイプで使用して、情報を ERP に転送できます。

- メッセージング・アダプタ
  - MQSeries
  - MSMQ
- バッチ・インターフェイス
  - インターフェイス・テーブル
  - 電子データ交換（EDI）
  - テーブル変換
  - 拡張プランニング・エージェント(APAg)/インテグレーション

参照

- ❑ インターフェイス・テーブルのフォーマットについては、『インタオペラビリティ』ガイドの「インターフェイス・テーブル」

Z トランザクションの処理

次のリストに、Z トランザクションを使用して ERP に情報を転送するために必要なタスクを示します。

- トランザクションに命名します。
- 受信インターフェイス・テーブルにレコードを追加します。
- 更新処理を実行します。
- エラーを確認します。
- 更新を確認します（オプション、ユーザー指定）。
- インターフェイス・テーブルからデータを除去します。

トランザクションの命名

Z トランザクション・タイプは、ユーザー定義コード 00/TT で定義されます。新規トランザクションを作成する場合は、そのトランザクションをユーザー定義コード 00/TT 内で定義する必要があります。新規トランザクション・タイプには、JDE で始まる長さ 8 文字以内の名前を指定する必要があります。次の例は、正しいトランザクション名を示しています。

- JDERR：入荷工程処理トランザクション
- JDEWO：作業オーダー見出しトランザクション

受信インターフェイス・テーブルへのレコードの追加

トランザクションを該当するインターフェイス・テーブルに書き込むと、その情報は処理のために ERP で使用可能になります。Z トランザクションは、ERP データベース・フォーマットになっているインターフェイス・テーブルに直接書き込むことができます。次のリストに、受信インターフェイス・テーブルにレコードを追加する方法を示します。

- フラット・ファイルを作成し、そのデータをインターフェイス・テーブル内のレコードに変換します。
- ERP のパブリッシュ API を使用してインターフェイス・テーブルを更新し、API（アプリケーション・プログラミング・インターフェイス）を記述します。
- 電子データ交換(EDI)を使用してインターフェイス・テーブルを更新します。
- メッセージを MQSeries または MSMQ メッセージング・アダプタに置きます。
- 構造化照会言語(SQL)または格納プロシージャを使用します。レコードを J.D. Edwards インターフェイス・テーブル・フォーマットに変換できる必要があります。

注意：

フラット・ファイルを使用して J.D. Edwards インターフェイス・テーブルにレコードを追加する場合は、作成しようとするトランザクション用のバージョンが〈フラット・ファイル変換(受信)〉プログラム(R47002C)に存在するかどうかを確認してください。

正味変更の考慮事項

トランザクションのデータ要素セットを新しい値に（ブランクの値を含めて）変更することは、正味変更（ネットチェンジ）処理と言われます。

正味変更は、故意に NULL を使用することによって処理されます。フィールドの NULL 値は、この受信フィールドが現在の値から変更されていないことをマスター・ビジネス関数に指示するものです。受信フィールドのその他の値は検証された後、データベースを更新します。

レコードがデータベースに挿入される場合、NULL であるフィールドはすべて1つのブランクに初期化されます。未編集トランザクション・テーブルでは、データベース・ミドルウェアが NULL 値をブランクに初期化しないようなその他の特性が必要になります。この機能により、テーブル・カラムに NULL 値を指定できます。この NULL 値はマスター・ビジネス関数データ構造体パラメータに割り当てることができます。

更新処理の実行

Z トランザクションは次のいずれかの方法で処理できます。

- 入力バッチ処理を実行します。これにより、多数のトランザクションをインターフェイス・テーブルに入れてから、バッチ・モードで一度にすべてのトランザクションを処理できます。
- サブシステム・ジョブを実行します。これにより、トランザクションを一度に 1 つずつ ERP に送信し、個々の完了を待たずにサブシステムを使用して処理を続行できます。

一部のアプリケーションの場合は、入力バッチ処理と入力サブシステム処理が用意されています。

参照

- 入力プロセッサのバッチ処理と入力サブシステムのバッチ処理については、『インタオペラビリティ』ガイドの「インタオペラビリティ・インターフェイス・テーブル情報」

入力バッチ処理の実行

入力バッチ処理を使用すると、1 つまたは複数のレコードをインターフェイス・テーブルに入れてから、UBE を実行して一度にすべてのレコードを処理できます。受信インタオペラビリティ処理をサポートしているアプリケーションの場合は、Solution Explorer を使用して入力バッチ処理を開始します。入力バッチ処理を選ぶと、レポート機能のバージョン・リストが表示されます。レポートの既存バージョンをそのまま使用したり、既存バージョンを変更したり、またはバージョンを追加できます。レポート・バージョンを使用する場合は、処理オプションとデータ選択を変更できます。入力バッチ処理プログラムにより、処理されたトランザクション、処理されたトランザクションの総数、処理中に発生したエラーをリスト表示する監査レポートが生成されます。

▶ 入力バッチ処理を実行するには

受信トランザクション・タイプに対して、入力バッチ処理を開始します。たとえば、〈購買オーダー〉プログラムには入力バッチ処理があります。入力バッチ処理が用意されているプログラムについては、この項の末尾の「インタオペラビリティ・インターフェイス・テーブル情報」を参照してください。

1. 〈バッチ・バージョンの処理〉で、処理オプションを検討して〈編集/更新（受信）〉プログラムがデータを処理する方法を確定します。
2. 実行したいバージョンを選んで［選択］をクリックします。
3. 〈バージョン・プロンプト〉で、次のいずれかをクリックし、レポート機能オプションを検討します。
   - Data Selection（データ選択）
   - Data Sequencing（データ順序設定）
4. ［投入］ボタンをクリックして、受信トランザクション・タイプに対する入力バッチ処理を開始します。

サブシステム・ジョブの実行

サブシステム・ジョブとは、データ待ち行列からレコードを処理し、ユーザーがジョブを終了するまで実行される連続ジョブです。サブシステム・ジョブは、サブシステム・テーブルに対して一度に 1 レコードを読み取って情報を取り出すことにより、各レコードに対して UBE またはテーブル変換を実行します。これにより、特定のキーを処理する〈受信プロセッサ〉バッチ処理がトリガーされます。必要に応じて、〈受信プロセッサ〉バッチ処理からプリプロセッサ・ビジネス関数を実行し、元のアプリケーション・レコードをインターフェイス・テーブル・レコードと照合するキー情報、たとえば現金入金確認または入荷確認のキーを設定します。最後のレコードを処理した後、サブシステム・ジョブは終了する代わりに特定の期間だけ待機してから新規レコードを取得しようとします。各サブシステム・ジョブに対して、サブシステム・テーブルに複数のレコードが存在する場合があります。

サブシステム・ジョブをスケジュールできます。

サブシステム・ジョブを開始するには、次の方法があります。

ビジネス関数の使用

受信トランザクションの場合は、汎用サブシステム・ビジネス関数 Add Inbound Transaction to Subsystem Queue(B0000175)を使用できます。この関数はレコードをサブシステム・ジョブ・マスター(F986113)に書き込み、サブシステムに認識すべきバッチ処理を指定します。また、ビジネス関数はキー・データをサブシステム・データ待ち行列に渡します。次に、トランザクションの処理を開始します。

Solution Explorer の使用

受信インタオペラビリティ処理をサポートしているアプリケーションの場合は、Solution Explorer を使用して入力サブシステムのバッチ処理を開始できます。サブシステム・ジョブは、通常のバッチ・ジョブと同じように開始します。他のバッチ・ジョブとは異なり、サブシステム・ジョブを実行できるのはサーバー上のみです。処理の前に ERP は、特定のサーバーのサブシステム・ジョブに対する制限を超えていないかどうかを確認します。制限を超えていると、サブシステム・ジョブは処理されません。Z トランザクションをほぼリアルタイム・モードで処理するには、システムの起動時にサブシステムを開始します。トランザクションをインターフェイス・テーブルに書き込む前に、要求をデータ待ち行列に入れる必要があります。

注意:

サブシステム・ジョブは、レコードの処理が完了した後に終了する代わりに、データ待ち行列内の新規データをチェックします。サブシステム・ジョブは、終了するまで実行されています。

参照

『システム・アドミニストレーション』ガイドの次のトピックを参照してください。

- サブシステムの作成と開始については「ERP 9.0 サブシステムの有効化」
- サブシステム・ジョブの停止と終了については「ERP 9.0 サブシステムの終了」
- サブシステム・ジョブのスケジュールリングについては「Scheduler アプリケーション」

▶ サブシステム・ジョブを実行するには

受信トランザクション・タイプに対して、〈受信プロセッサ〉バッチ処理を開始します。たとえば、〈購買オーダー〉プログラムには入力サブシステム処理があります。サブシステム・バッチ処理が用意されているプログラムについては、この項の末尾の「インタオペラビリティ・インターフェイス・テーブル情報」を参照してください。

処理オプションを検討し、サブシステム・ジョブがデータを処理する方法を確定します。

1. 〈バッチ・バージョンの処理–使用可能なバージョン〉フォームで、実行するサブシステムのバージョンを選んで[選択]をクリックします。
2. 〈バージョン・プロンプト〉で、次のいずれかをクリックし、レポート機能オプションを検討します。
   - データ選択
   - データ順序
3. [投入]ボタンをクリックします。

エラーの有無のチェック

入力バッチ処理は、インターフェイス・テーブルのデータを使用して、ビジネス・ロジックで示された該当する J.D. Edwards アプリケーション・テーブルを更新します。トランザクションにエラーが発生すると、〈プロセッサ監査証跡〉〉レポートでレコードにフラグが設定され、〈従業員ワークセンター〉にアクション・メッセージ形式でエラー・メッセージが送信されます。これらのアクション・メッセージを呼び出すと、改訂アプリケーションが呼び出され、インターフェイス・テーブルを修正できます。

ワークセンターのエラーを検討する際、インターフェイス・テーブルの関連トランザクションに直接リンクして、修正を行うことができます。改訂アプリケーションを使用して、即時処理をするために修正された個々のトランザクションを待ち行列に再投入するか、またはすべてのトランザクション・エラーを修正し、バッチ処理で一度にトランザクションを再投入します。

既存ファイルを正常に更新したトランザクションには、インターフェイス・テーブルに正常に処理されたことを示すフラグが設定されます。

参照

- ❑ インターフェイス・テーブル内のレコードの修正については、『インタオペラビリティ』ガイドの「改訂アプリケーション」

更新の確認

この手順は任意です。ビジネス関数を使用すると、J.D. Edwards システムに送信したトランザクションの処理が完了したことを通知する確認関数を提供できます。処理が完了すると、処理状況をユーザーに通知するためにその要求で指定されている関数が呼び出されます。確認関数はユーザーのスペックに書き込まれますが、J.D. Edwards 定義のデータ構造体を使用する必要があります。インタオペラビリティ受信確認機能は、Call Vendor – Specific Function – Inbound ビジネス関数を介して〈受信プロセッサ〉バッチ処理から呼び出されます。

確認関数は各プロセスに特有のものであり、次のパラメータを受け入れる必要があります。

User ID	11 文字
バッチ No.	16 文字
トランザクション番号	23 文字
行 No.	2 文字
処理状況	1 文字

最初の 4 パラメータは、処理済みトランザクションに対するキー（EDUS、EDBT、EDTN、EDLN）です。関数を含むライブラリの最後のフル・パスを、トランザクションを処理するサブシステム・バッチ処理に渡す必要があります。このデータは、受信トランザクション・サブシステム・データ構造体から渡されます。

サブシステム・バッチ処理がトランザクション処理を終了した後、受信確認関数を呼び出し、処理されたトランザクションに対するキーおよびトランザクションが正常に処理されたかどうかの通知を渡します。関数にはトランザクションの成功または失敗に基づいて適切な処理がなされるようにロジックを含めます。

トランザクション確認の関数を作成する場合、その関数を使用して次のタスクを実行できます。

当初トランザクションの更新	当初トランザクションとインタオペラビリティ・テーブルに書き込まれたトランザクションとの相互参照を作成することで、当初トランザクションにアクセスして、計算された完了済みまたはエラーの状況フラグを更新できます。 この関数に戻されるキーを使用し、インタオペラビリティ・インターフェイス・テーブルに書き込まれたトランザクションにアクセスして、計算されたデータまたはデフォルト指定されたデータを取り込み、当初トランザクションを更新できます。
J.D. Edwards 以外の他のビジネス処理の実行	トランザクションが完了した場合、J.D. Edwards 以外のソフトウェアでトランザクションを完了するビジネス処理を実行できます。
ユーザーへのメッセージ送信	ユーザーに当初トランザクションの状況を通知できます。

インターフェイス・テーブルのデータ除去

インターフェイス・テーブルから ERP データベースに対して正常に更新されたレコードを、定期的に除去する必要があります。

参照

『インタオペラビリティ』ガイドの次のトピックを参照してください。

- 除去バッチ処理については「インタオペラビリティ・インターフェイス・テーブル情報」
- インターフェイス・テーブルからデータを除去する方法については「インターフェイス・テーブル情報の除去」

フラット・ファイル

フラット・ファイルはユーザー定義フォーマットとも呼ばれ、通常ワークステーションまたはサーバー上に保管されたテキスト・ファイルであり、一般的には ASCII 文字セットが使用されます。フラット・ファイル内のデータは、連続した情報の文字列として保管されているため、リレーショナル・データベース・テーブルのようなリレーションシップは定義されていません。フラット・ファイルを使用すると、対話方法が他にないアプリケーションからデータをインポートまたはエクスポートできます。たとえば、ERP と他のアプリケーション間でデータを共有したい場合があります。ERP 以外のアプリケーションが ERP がサポートしているデータベースをサポートしていない場合、フラット・ファイルを使用しなければ、2 つのアプリケーション間でデータを転送することはできません。

フラット・ファイルは次のモデル・タイプで使用できます。

- XPI
  - エンタープライズ XPI
  - インターエンタープライズ XPI

  注:
  一般的に、XPI ではフラット・ファイル処理にファイル入出力アダプタが使用されます。

- バッチ・インターフェイス
  - インターフェイス・テーブル
  - 電子データ・インターフェイス（EDI）
  - テーブル変換

フラット・ファイルを使用して ERP にデータを転送する場合、データで既存データベースを更新するには、J.D. Edwards フォーマットに変換する必要があります。J.D. Edwards インターフェイス・テーブルを変換プログラム、電子データ・インターフェイス(EDI)、またはテーブル変換と共に使用して、フラット・ファイル・データをフォーマットできます。EDI またはテーブル変換を使用すると、フラット・ファイルへの入力となる ERP データを取得できます。

XPI と、バッチ抽出プログラムのような一部の J.D. Edwards バッチ・インターフェイスは、フラット・ファイルを受け入れて情報をデータ・フォーマットに解析する機能があります。

参照

- EDI については『Electronic Data Interchange（電子データ交換）』ガイドの「Data Interface（データ・インターフェイス）」
- テーブル変換については『テーブル・コンバージョン』ガイドの「テーブル変換の設定」

『インタオペラビリティ』ガイドの次のトピックを参照してください。

- カスタム・インターフェイス・テーブルの作成については、「インターフェイス・テーブル」
- J.D. Edwards インターフェイス・テーブル用のプロセス・リストについては、「インタオペラビリティ・インターフェイス・テーブル情報」

フラット・ファイルのフォーマット

J.D. Edwards インターフェイス・テーブルを使用してデータをインポートする場合、フラット・ファイルのフォーマットはユーザー定義でも文字区切でもかまいません。次の例に、5 つのカラム Last、First、Addr、City、Phone を持つユーザー定義フォーマットの単一のデータベース文字レコードを示します。

Doe	John	123 Main	Any town	5551234	←・データベース・レコード
Last	First	Addr	City	Phone	

上記のユーザー定義フォーマットの例は固定幅カラム・フォーマットで、各カラムごとのデータはすべて、各ローの同じ相対位置から始まります。

次の例は、同じデータを文字区切フォーマットで示したものです。

“Doe”, “John”, “123 Main”, “Anytown”, “5551234”

フラット・ファイル変換要件の設定

フラット・ファイルのレコード・フォーマットは、インターフェイス・テーブルのフォーマットに準じる必要があります。つまり、テーブルのすべてのカラムは、フラット・ファイル・レコードに入れられ、インターフェイス・テーブルと同じ順にカラムが表示される必要があります。インターフェイス・テーブルのフィールドは、ブランクであっても、すべて書き込まれる必要があります。各フィールドは、開始と終了を示す記号で囲みます。一般的に、この記号は二重引用符(“”)です。また、各フィールドはフィールド区切り文字で隣のフィールドと区切ります。一般的に、この区切り文字はカンマ(,)です。ただし、フィールドの解釈で支障がない限り、どのフィールド区切り文字およびどのテキスト修飾子も使用できます。変換プログラムの処理オプションを設定して、テキスト修飾子とフィールド区切り文字を定義します。小数点のある伝票を受信する場合、プレースホルダー(ピリオドなど)を使用して小数点の位置を示す必要があります。ユーザー優先情報テーブルにプレースホルダーを定義します。

フラット・ファイル・レコードの最初のフィールド値は、レコード・タイプを示します。つまり、最初のフィールド値は、変換プログラムでレコードを挿入する必要のあるインターフェイス・テーブルを示します。レコード・タイプ値は、ユーザー定義コード(00/RD)によって定義され、保管されています。ハード・コード化された値は次のとおりです。

1 - 見出し

2 - 明細

3 - その他の見出し

4 - 追加明細

5 - SDQ

6 - 住所

7 – 見出しテキスト

8 – 詳細テキスト

たとえば、見出しテーブルのレコードは次のようになります(この例では、テーブル・レイアウト標準は無視されます)。

レコード・タイプ	Name(名前)	住所	City(市)	郵便番号
1	Joe	<Blank>	Denver	80237

フラット・ファイルのレコードは、次のようになります。

"1","Joe"," ","Denver","80237"

見出しレコード・タイプに対応する"1"、住所カラムの<Blank>に対応するスペースに注意してください。

日付は MM/DD/YY のフォーマットでなければなりません。数値フィールドは、位取りとして小数点を指定する必要があります。カンマは使用できません。

フラット・ファイルの変換

J.D. Edwards では、Windows プラットフォーム上でのフラット・ファイル変換のみをサポートしています。Windows プラットフォームでは、〈受信フラット・ファイル変換〉プログラム(R74002C)または Import Flat File To JDE File ビジネス関数(B4700240)を使用できます。

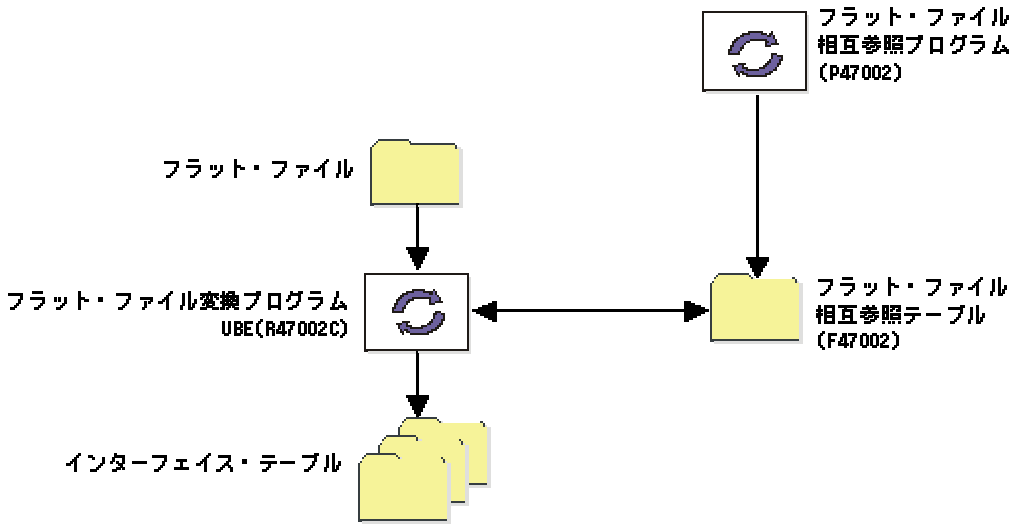
フラット・ファイル変換プログラム

Windows プラットフォームでは、〈受信フラット・ファイル変換〉プログラム(R47002C)を使用して、フラット・ファイルを J.D. Edwards のインターフェイス・テーブルにインポートできます。各インターフェイス・テーブルに対して〈受信フラット・ファイル変換〉プログラム(R47002C)に個別のバージョンを作成します。

注:

〈受信フラット・ファイル変換〉プログラムを使用するには、PC 上のドライブをフラット・ファイルの位置にマップする必要があります。

次の図に、フラット・ファイルを使用して J.D. Edwards のインターフェイス・テーブルを更新するプロセスを示します。



〈フラット・ファイル相互参照〉プログラム(P47002)を使用して、フラット・ファイル相互参照テーブル(F47002)を更新します。この変換プログラムでは、フラット・ファイル相互参照テーブル(F47002)を使用し、受け取るトランザクション・タイプに基づいて、どのフラット・ファイルから読み取るかが判別されます。次のリストに、フラット・ファイル相互参照テーブル(F47002)内の情報を示します。

[Transaction Type]	特定のトランザクション・タイプ。トランザクション・タイプはユーザー定義コード(00/TT)内で定義する必要があります。
Direction Indicator (送受信インジケータ)	トランザクションの方向(送信/受信)を指定するコード。送受信インジケータはユーザー定義コード(00/DN)内で定義する必要があります。
Flat File Name (フラット・ファイル名)	Windows PC 上のフラット・ファイルへのパス。
Record Type (レコード・タイプ)	トランザクション・レコードを見出し、明細などとしてマークする識別子。レコード・タイプ・インジケータはユーザー定義コード(00/RD)内で定義する必要があります。
File Name(ファイル名)	有効な ERP インターフェイス・テーブル。

変換プログラムはフラット・ファイル相互参照テーブル(F47002)を使用して、フラット・ファイルを ERP インターフェイス・テーブルに変換します。このプログラムでは、読取り元となるフラット・ファイルと、そのフラット・ファイル内のレコード・タイプの両方が認識されます。各フラット・ファイルには、対応するインターフェイス・テーブルのレコードに基づいた、長さが異なるレコードが含まれます。

変換プログラムはフラット・ファイルの各レコードを読み取り、フラット・ファイルに指定したテキスト修飾子およびフィールド区切り文字に基づいて、インターフェイス・テーブルの各フィールドにそのレコード・データをマップします。変換プログラムが各フィールドを正常に解釈し、該当する受信インターフェイス・テーブル内の対応するフィールドに移動するには、すべてのフィールドが正しくフォーマットされている必要があります。

また、このプログラムは、フィールド・データを 1 つの完全なレコードとして、インターフェイス・テーブルに挿入します。データの変換中にエラーが発生すると、インターフェイス・テーブルは更新されません。フラット・ファイルはサードパーティ・ソフトウェアによって作成された外部オブジェクトであるため、変換プログラムはどのフラット・ファイル・データ・フィールドのフォーマットが正しくないかを判別でき

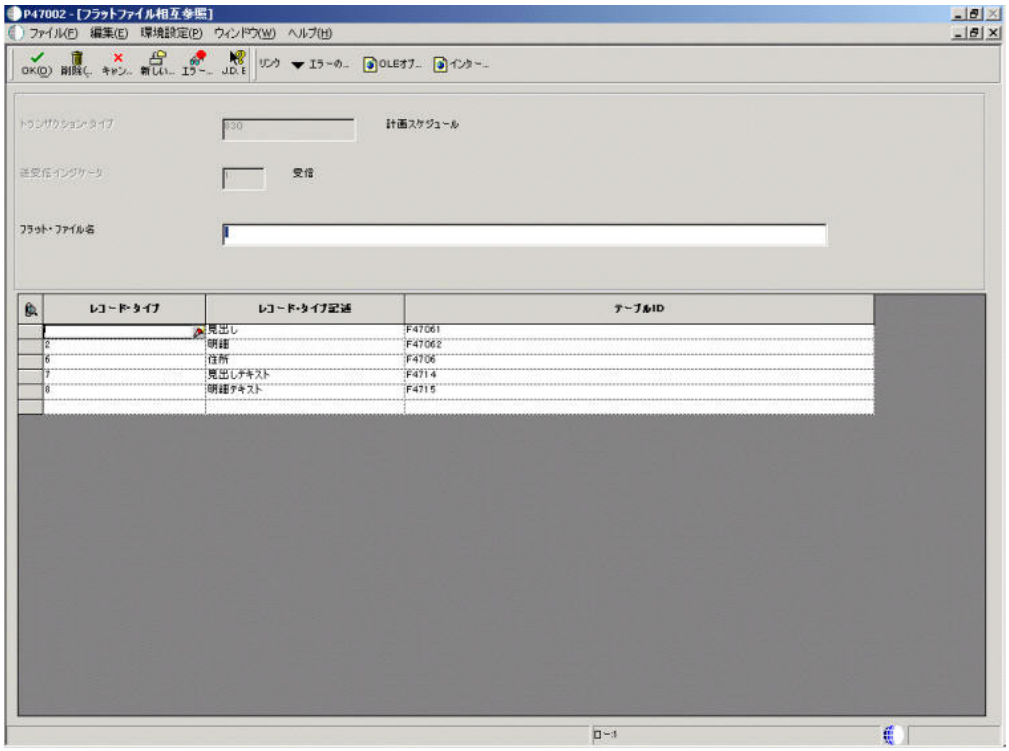
ません。そのため、フラット・ファイル内に誤りがある場合は、それを判別する必要があります。変換プログラムは、フラット・ファイルのデータをすべてインターフェイス・テーブルへと正常に変換した場合は、変換後にフラット・ファイルを自動的に削除します。変換プログラムの次のプロセスを開始するように処理オプションを設定している場合は、データが正常に変換された後、そのインターフェイス・テーブルに対して〈受信プロセッサ〉バッチ処理が自動的に実行されます。

フラット・ファイルが正常に処理されなかった場合は、〈ワークフロー管理〉メニュー(G02)から〈従業員ワークセンター〉にアクセスしてエラーを確認できます。エラー条件を修正した後、〈受信フラット・ファイル変換〉プログラム(R47002C)を再実行します。

▶ フラット・ファイル相互参照を更新するには

フラット・ファイル変換をサポートしているアプリケーションから、〈フラット・ファイル相互参照〉プログラム(P47002)を開きます。

1. 〈フラットファイル相互参照の処理〉で、次のフィールドに値を入力します。
   - トランザクション
2. [ロー]メニューから[定義]を選びます。



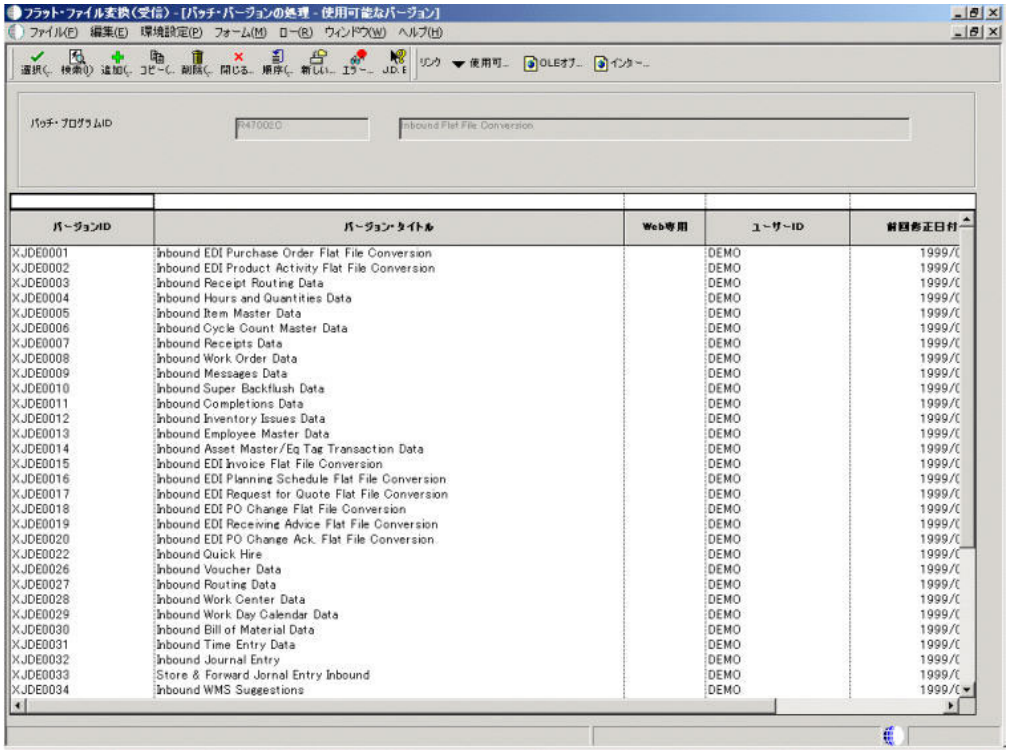
3. 〈フラット・ファイル相互参照〉で、見出し領域の次のフィールドに値を入力します。
   - フラット・ファイル名
4. グリッドで、必要に応じて次のフィールドを変更します。
   - レコード・タイプ
   - レコード・タイプ記述
   - ファイル名

フィールド記述

記述	用語解説
フラット・ファイル名	フラット・ファイルのファイル名。これにはフラット・ファイルが存在するディレクトリのパスを含みます。
レコード・タイプ	EDI トランザクション・レコードが見出しと明細情報のどちらかを識別するコード。これは、EDI でのみ使う機能です。
レコード・タイプ記述	ユーザー定義名称または備考。
テーブル ID	テーブルを識別するための番号。たとえば、勘定科目マスターは F0901 です。命名規則については「プログラミング標準マニュアル」を参照してください。

► フラット・ファイルからインポートするには

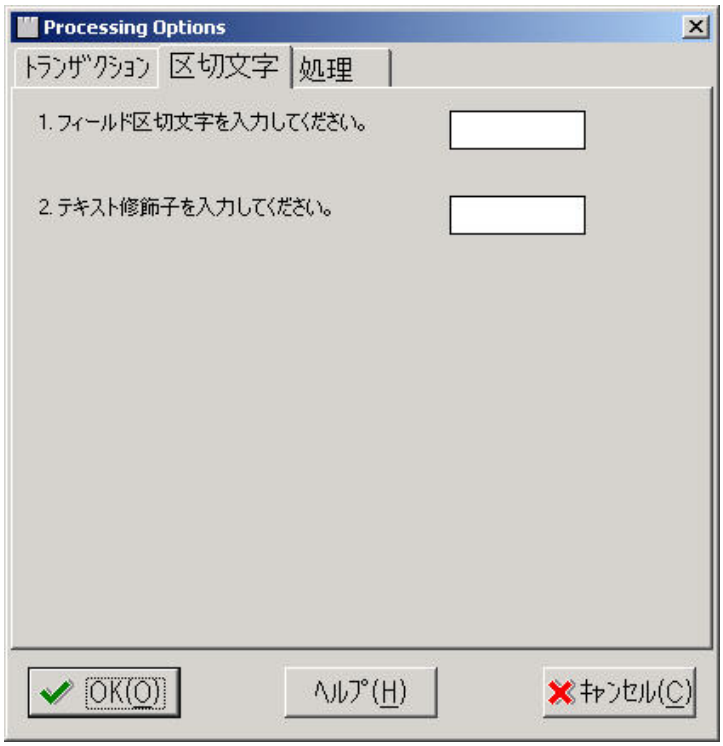
〈フラット・ファイル変換〉バッチ処理(R47002C)を開きます。



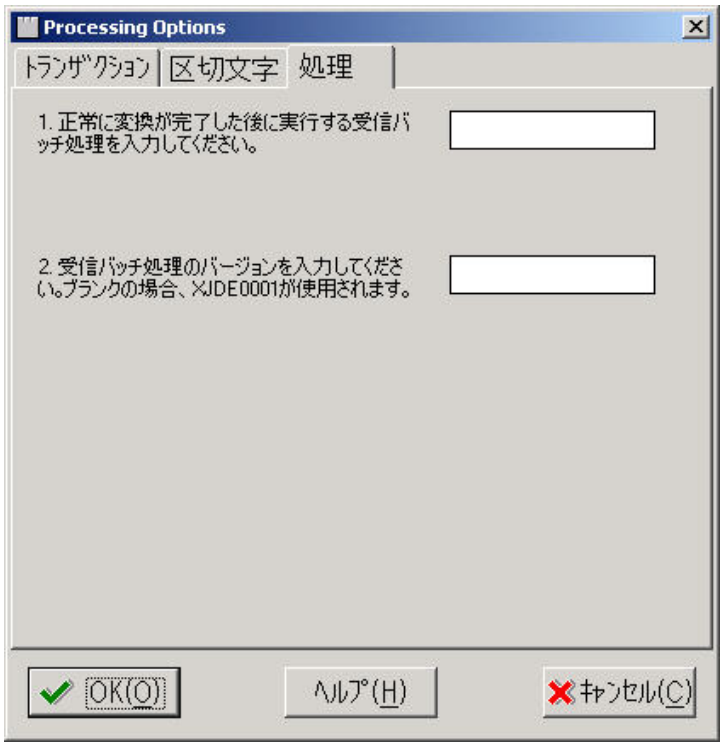
1. 〈バッチ・バージョンの処理 - 使用可能なバージョン〉で、使用したいプログラム・バージョンを選びます。

2. [ロー]メニューから[処理オプション]を選びます。

3. [トランザクション]タブをクリックし、インポートしているトランザクション・タイプを入力します。たとえば"850"です。



4. [区切り文字]タブをクリックし、フィールドおよびテキストの識別に使われるフィールド区切り文字およびテキスト修飾子文字を入力します。



5. [処理]タブをクリックし、フラット・ファイル変換プログラムが正常に終了した後、実行するプログラムのバージョンおよび受信プログラム名を入力します。
6. [OK]をクリックします。
7. 〈バッチ・バージョンの処理〉で、[選択]をクリックします。
8. 〈バージョン・プロンプト〉で、次のいずれかをクリックしてレポート機能オプションを検討します。
   - データ選択
   - データ順序
9. [投入]ボタンをクリックします。

ビジネス関数を使用したフラット・ファイルのインポート

Windows プラットフォームでは、Import Flat File To JDE File ビジネス関数(B4700240)を使用できます。サーバーのオペレーティング・システムに変更が加えられている場合や、オペレーティング・システムによってはファイルの保管方法が異なる場合があるため、J.D. Edwards では Windows プラットフォームで実行されるビジネス関数のみをサポートしています。Import Flat File To JDE File ビジネス関数(B4700240)を使用する場合は、次の制約に注意してください。

- [トランザクション・タイプ]および[フラット・ファイル名]フィールドは必須フィールドです。
- [レコード・タイプ]フィールドに入力できるのは 1 文字のみです。
- 各行の最大長は 4095 文字です。
- 最大レコード・タイプ数は 40 です。

- すべての行で 1 つずつレコードを指定する必要があります。
- カラム区切り文字と同じテキスト修飾子は使用できません。

ERP インターフェイス・テーブルへの挿入前にフラット・ファイル・データのフォーマットが正しいかどうかを確認するために、ビジネス関数はプライマリ・インデックス詳細テーブル(F98713)を使用して、プライマリ・インデックス・キー情報を取得します。通常、プライマリ・インデックス詳細テーブル(F98713)は、〈オブジェクト構成マネージャ〉の[デフォルト・ビジネス・データ]テーブル・マッピングの下にあります。ビジネス関数でプライマリ・インデックス詳細テーブル(F98713)を検出できるように、次のどちらかの操作を実行する必要があります。

- システム・データ・ソース内でプライマリ・インデックス詳細テーブル(F98713)をマップします。
- プライマリ・インデックス詳細テーブル(F98713)がビジネス・データ・ソース内に存在することを確認します。

システム・データ・ソース内でのプライマリ・インデックス詳細テーブル(F98713)のマッピング

システム・データ・ソース内でテーブルをマップするには、プライマリ・インデックス詳細テーブル(F98713)を指す OCM マッピングをセントラル・オブジェクト・データ・ソースに追加します。

プライマリ・インデックス詳細テーブル(F98713)がビジネス・データ・ソース内に存在するかどうかの確認

プライマリ・インデックス詳細テーブル(F98713)をビジネス・データ・ソース内で生成する場合は、そのファイル拡張子が PC 上で非表示に設定されていることを確認する必要があります。ファイル拡張子を非表示にする手順は、次のとおりです。

1. [スタート]/[設定]/[コントロール パネル]/[フォルダ オプション]を順に選んで[表示]タブをクリックします。
2. [登録されているファイルの拡張子は表示しない]オプションをオンにして[OK]をクリックします。

フラット・ファイル相互参照テーブル(F47002)の[フラット・ファイル名]フィールドにファイル拡張子が指定されていることも確認する必要があります。たとえば、C:¥flatfiles¥850.txt などです。

フラット・ファイル変換のエラー・メッセージ

ビジネス関数を使用してフラット・ファイルを変換する際に、次の 2 つのエラーが発生することがあります。

- 4363 Null Pointer(NULL ポインタ)
- 4377 Invalid Input Parameter(無効な入力パラメータ)

どちらのエラーも、ビジネス関数内部の問題です。

次のエラーは、ユーザー設定の問題や CNC インプリメンテーションの問題が原因で発生することがあります。

- 0073 Invalid File Name(無効なファイル名)
- 128J (filename) Insert Failed((ファイル名)の挿入に失敗)
- 3003 Open of File Unsuccessful(ファイルのオープンに失敗)
- 4569 Invalid Format(無効なフォーマット)

フラット・ファイル API

J.D. Edwards には、既存のフラット・ファイル API に加えて、非 Unicode フラット・ファイル用の API が用意されています。Unicode API が必要になるのは、フラット・ファイル・データが ERP 外部のプロセスによって読み書きされる場合です。jdeFWrite()や jdeFRead()などの J.D. Edwards API は、フラット・ファイル・データを変換しません。つまり、文字データに関するデフォルトのフラット・ファイル I/O は Unicode です。ERP で生成されたフラット・ファイルを使用する場合に、受信側システムが Unicode データを予期していなければ、フラット・ファイルを正常に読み取ることができません。たとえば、受信側システムが Unicode 対応ではなく、日本語 Shift_JIS コード・ページ（またはエンコーディング）によるデータを予期している場合、フラット・ファイルを正常に読み取ることはできません。日本語 Shift_JIS ページによるフラット・ファイルを作成可能にするには、〈Unicode フラット・ファイル・エンコード構成〉プログラム(P93081)を使用して、フラット・ファイルを作成するアプリケーションを構成する必要があります。フラット・ファイルがワークテーブルまたはデバッグ・ファイルで、ERP でのみ読み書きされる場合は、既存のフラット・ファイル API を使用してください。たとえば、ビジネス関数がフラット・ファイル内である種のキャッシュ操作を実行する場合、そのフラット・ファイルのデータを変換する必要はありません。

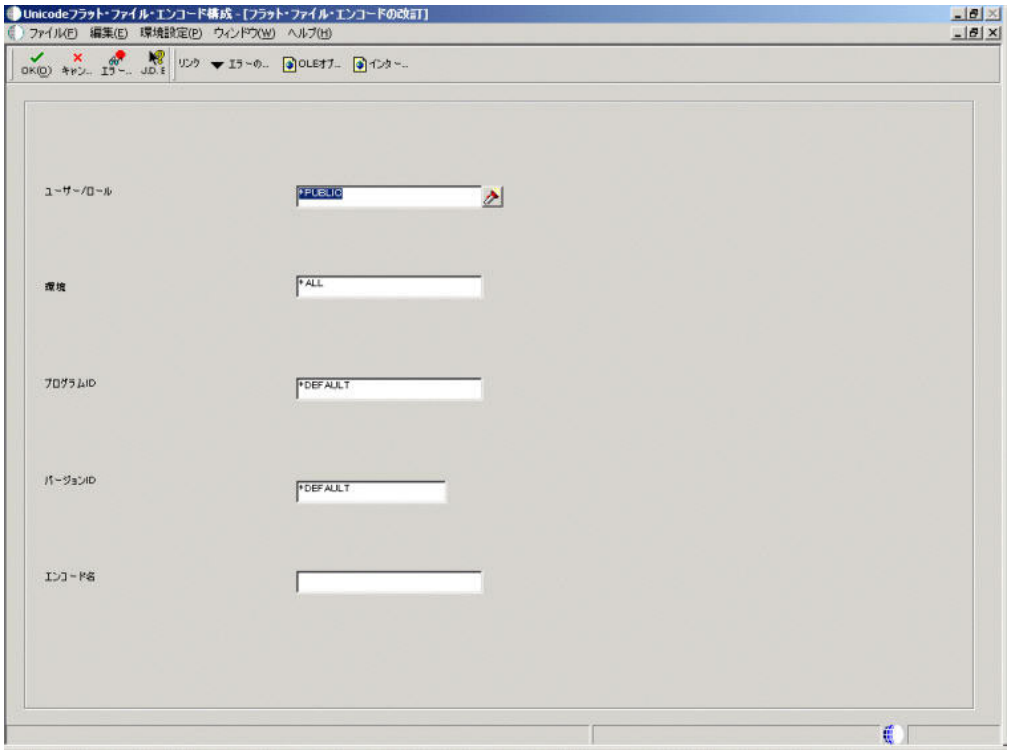
ERP による Unicode への変換では、すべての文字データ表現にメモリ内の UCS-2 エンコード、つまり 1 文字(JCHAR)当たり 2 バイトが使用されます。出力フラット・ファイル API に渡される文字データは、JCHAR(UCS-2)にする必要があります。入力フラット・ファイル API は、構成済みコード・ページの文字データを UCS-2 に変換し、JCHAR（または JCHAR 文字列）で戻します。フラット・ファイル変換 API を使用すると、実行時にフラット・ファイル用のコード・ページを構成できます。フラット・ファイルのコード・ページ設定には、〈Unicode フラット・ファイル・エンコード構成〉プログラム(P93081)を使用します。フラット・ファイル・エンコードは、アプリケーション名、アプリケーション・バージョン名、ユーザー名、および環境名などの属性に基づいています。

構成アプリケーションでコード・ページを指定しなければ、API は特定の関数への入力データを介してフラット・ファイル I/O の受け渡しを実行します。たとえば、jdeFWriteConvert()は Unicode データを書き込み、変換は実行されません。

▶ フラット・ファイル・エンコードを設定するには

〈システム・アドミニストレーション・ツール〉メニュー(GH9011)で、〈Unicode フラット・ファイル・エンコード構成〉を選びます。

1. 〈フラット・ファイル・エンコードの処理〉で、[追加]をクリックします。



2. 〈フラット・ファイル・エンコードの改訂〉で、次のフィールドに値を入力します。
   - ユーザー/ロール
   - 環境
   - プログラム ID
   - バージョン ID
   - エンコード名
3. [OK]をクリックして入力を保存します。
4. [キャンセル]をクリックして〈フラット・ファイル・エンコーディングの処理〉に戻ります。
5. 〈フラット・ファイル・エンコーディングの処理〉で、[検索]をクリックします。
6. 新規エントリをアクティブ化するには、グリッドでその入力を選びます。
7. [ロー]メニューから[変更状況]を選びます。

   アクティブ状況は AV、非アクティブ状況は NA です。
8. [閉じる]をクリックしてメイン・メニューに戻ります。

フィールド記述

記述	用語解説
ユーザー/ロール	セキュリティを目的とし、ユーザーをグループに分類するために使用されるプロファイル。グループ・プロファイルは、グループ・メンバーに特定のプログラムへのアクセス権を与えるのに使用します。 ユーザー・クラス・グループの作成ルールは次のとおりです。 クラス・グループ・プロファイルは、他のシステム・プロファイルとの衝突を避けるため、*で始めます。 ユーザー・クラス・グループ・フィールドは、新しいグループ・プロファイルの入力時にはブランクにします。
環境	インストール・アプリケーションの場合、環境名はプラン名とも呼ばれ、インストール/再インストールのための環境を識別します。環境またはバージョン・アプリケーションの場合、アプリケーションまたはバージョンのスペック・データの場所を識別するパス・コードです。
プログラム ID	対話型プログラムまたはバッチ・プログラムを識別する番号。たとえば、〈受注オーダー入力〉プログラムの ID は P4210（対話型）、〈請求書印刷〉プログラムの ID は R42565（バッチ）です。プログラム ID の桁数は固定ではなく、TSSXXX の形式で構造化されています。 T プログラム名の最初の英字が、P はプログラム、R はレポートなどそのタイプを示します。たとえば、P4210 の P は、これがプログラムであることを示します。 SS 2 番目と 3 番目は数字で、システム・コードを示します。たとえば、P4210 の 42 は、このプログラムがシステム 42（受注管理システム）に属していることを示します。 XXX プログラム名の残りの数字は固有のプログラムまたはレポートを示します。たとえば、P4210 の"10"は、受注オーダー入力アプリケーションであることを表します。
バージョン ID	アプリケーションやレポートの実行方法の指定に使用するユーザー定義のスペックです。バージョンを使用することで、ユーザー定義の処理オプション値やデータ選択、順序オプションなどをグループ化して保存します。対話型バージョンは（通常、タスク・レベルで）アプリケーションと関連付けられています。バッチ・バージョンはバッチ・プログラムまたはレポートと関連付けられています。バッチ・プログラムを実行する場合はバージョンを選択する必要があります。
エンコード名	フラット・ファイルを作成または使用するのにシステムで使用するエンコード名を示すコード。

参照

- フラット・ファイル・エンコードについては『システム・アドミニストレーション』ガイドの「フラット・ファイル・エンコードの処理」

イベント

J.D. Edwards のイベント機能は、ERP トランザクションをさまざまな方法でキャプチャできるインフラストラクチャを提供します。また、サードパーティ・ソフトウェア、エンドユーザー、および XPI、CRM、APS のような他の J.D. Edwards のシステムに、リアルタイムで通知を行います。

ERP の通知をイベントといいます。ERP イベント・システムでは、パブリッシュ/サブスクライブ・モデルをインプリメントしています。ERP イベントは、その詳細情報を含む XML ドキュメントの形式でサブスクライバに配信されます。たとえば受注オーダーが入力されたとき、受注オーダー情報は自動的に CRM またはサプライチェーン管理アプリケーションに送られ、次の処理に移されます。IBM システムの場合は、MQSeries メッセージ処理を使用してイベントを受信できます。MS システムの場合は、MSMQ メッセージ処理を使用してイベントを受信できます。MQSeries と MSMQ は、ERP との 2 点間インターフェイスを提供します。ERP は、次の 3 種類のイベントをサポートしています。

Z イベント	ERP インターフェイス・テーブル機能を使用して ERP トランザクションをキャプチャし、特定のトランザクションが発生した時点での通知を要求したサードパーティ・ソフトウェア、エンドユーザー、および他の J.D. Edwards システムに通知を提供するサービス。
「リアルタイム・イベント」	システム呼出しを使用して ERP トランザクションを発生時にキャプチャし、特定のトランザクションが発生した時点での通知を要求したサードパーティ・ソフトウェア、エンドユーザー、および他の J.D. Edwards システムに通知を提供するサービス。
XAPI イベント	システム呼出しを使用して ERP トランザクションを発生時にキャプチャし、指定のトランザクションが発生した時点での通知を要求したサードパーティ・ソフトウェア、エンドユーザー、および他の J.D. Edwards システムを呼び出して応答を戻すサービス。

ERP イベント・システムは、次のモジュールで構成されます。

- イベント・ディストリビュータ
- イベント・ジェネレータ
- トランスポート・ドライバ

イベント・ディストリビュータは、イベント通知（EVN）カーネルと呼ばれる ERP のカーネル・プロセスです。EVN カーネルは、サブスクライバを管理し、イベント発生時にサブスクライバに通知します。Z イベント、リアルタイム・イベント、および XAPI イベントが EVN カーネルを共有します。

イベント・ジェネレータとは、ERP イベントを生成する機能を持つプロセスまたはライブラリです。J.D. Edwards は、次の 3 つのデフォルト・イベント・ジェネレータを用意しています。

- Z イベント・ジェネレータ。Z イベントを生成します。
- リアルタイム・イベント・ジェネレータ。リアルタイム・イベントを生成します。
- XAPI イベント・ジェネレータ。XAPI イベントを生成します。

Z イベント、リアルタイム・イベント、および XAPI イベントの XML ドキュメントはそれぞれわずかながら異なっています。

イベント・ディストリビュータは、トランスポート・ドライバを使用して ERP イベントを送信します。ERP では、JDENET を使用するデフォルトのトランスポート･ドライバを用意しています。また、イベント・ディストリビュータは MQSeries または MSMQ トランスポート・ドライバを使用して、ERP イベント・ドキュメントを指定の MQSeries または MSMQ 送信待ち行列に送信することもできます。MQSeries または MSMQ トランスポート・ドライバを使用してイベントを受信する場合は、インタオペラビリティ・イベント定義テーブル(F90701)内で定義されているイベントをすべて受信することになります。

次のモデル・タイプでイベント生成がサポートされます。

- XPI
  - エンタープライズ XPI
  - インターエンタープライズ XPI
- コネクタ
  - 動的 Java コネクタ
  - Java コネクタ
  - COM コネクタ
- メッセージング・アダプタ
  - ❑ MQSeries 用アダプタ（Z イベントのみ）
  - ❑ MSMQ 用アダプタ（Z イベントのみ）

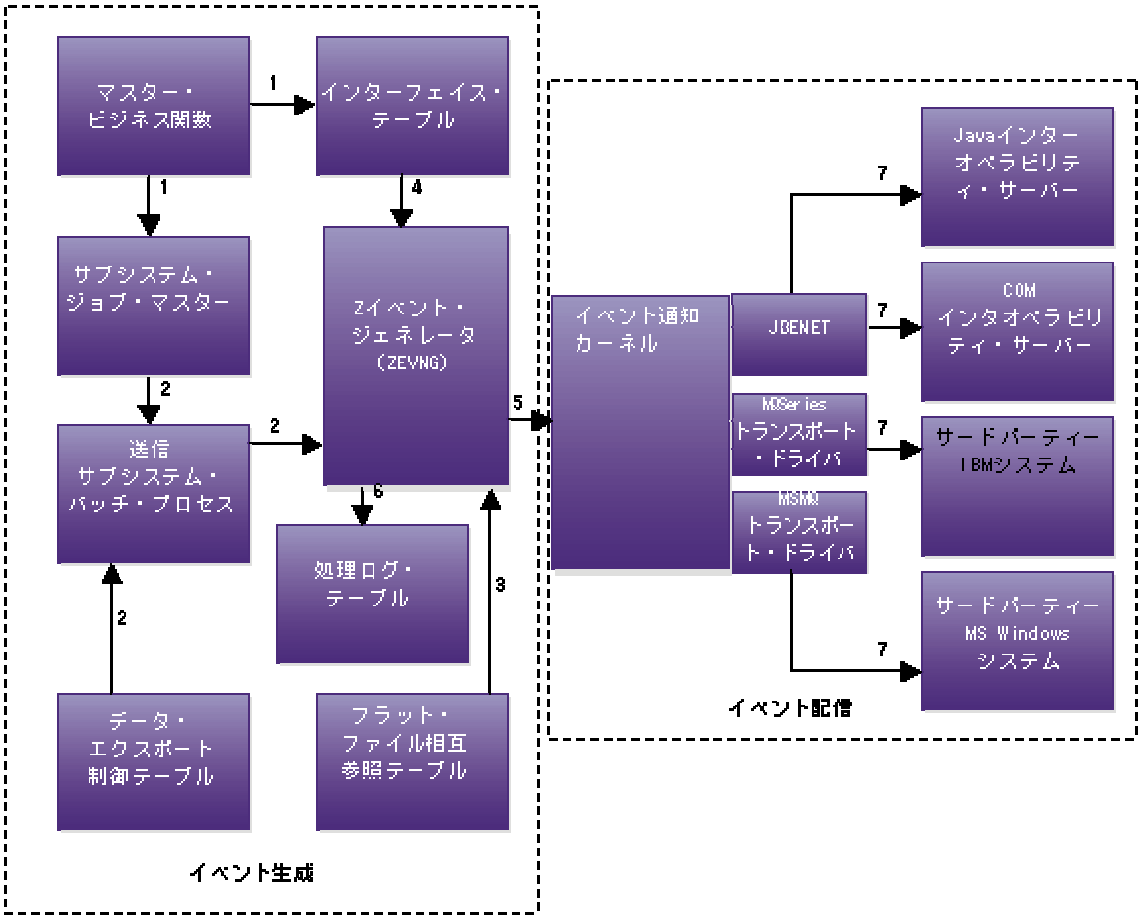
注:
MQSeries または MSMQ 待ち行列には、それぞれのトランスポート・ドライバを使用してすべてのイベントを送信できます。ただし、ERP からの送信トランザクションの場合、MQSeries および MSMQ アダプタでサポートされるのは Z イベントのみです。ERP への受信トランザクションは、この 2 つのアダプタでサポートされています。

はじめる前に

- ❑ ERP エンタープライズ・サーバーに対する ERP セキュリティを有効にする必要があります。
- ❑ ERP エンタープライズ・サーバーの jde.ini ファイルの[SECURITY]セクションで、デフォルト・ユーザーにセキュリティ・レコード（有効な ERP ユーザー）を設定する必要があります。

Z イベント

Z イベントは、インタオペラビリティ・トランザクションが発生したことを示す、ほぼリアルタイムの通知です。Z イベントを生成するために、ERP では Z イベント・ジェネレータと既存のインターフェイス・テーブルのインフラストラクチャが使用されます。既存の ERP インターフェイス・テーブルを使用できます。また、作成時に J.D. Edwards 標準に従っていれば、カスタマイズされたインターフェイス・テーブルを作成することも可能です。次の図に Z イベント生成用のプロセスとデータのロジックを示し、続いて説明します。



1. ERP トランザクションが発生すると、マスター・ビジネス関数(MBF)は、該当するインターフェイス・テーブルにトランザクション情報を書き込みます。また、サブシステム･ジョブ･マスター(F986113)に更新レコードを送ります。
2. バッチ・プロセスが、サブシステム・ジョブ・マスター(F986113)をモニタリングします。バッチ・プロセスは、サブシステム・ジョブ・マスター(F986113)内で W 状況を発見すると、Z イベント・ジェネレータに通知します。バッチ・プロセスはデータ・エクスポート制御テーブル(F0047)を参照し、どの Z イベント・ジェネレータを呼び出すかを確定します。
3. フラット・ファイル相互参照テーブル(F47002)は、トランザクションとレコードが保管されているインターフェイス・テーブル間の相互参照を提供します。この情報が Z イベント・ジェネレータで使用されます。
4. Z イベント・ジェネレータが、インターフェイス・テーブルからトランザクション情報を取り込み、J.D. Edwards DTD を使用して XML ドキュメントに変換します。
5. Z イベント・ジェネレータがイベントを XML ドキュメント形式でイベント通知カーネルに送信し、そこから配布されます。
6. イベントが正常に生成されると、処理ログ・テーブル(F0046)内で正常に生成されたカラムが更新されます。UBE が、処理ログ・テーブル(F0046)内の情報に基づいて、インターフェイス・テーブル内の情報を除去します。
7. イベント通知カーネルが、その XML ドキュメントをすべてのサブスクライバに送ります。

注:

MQSeries または MSMQ トランスポート・ドライバを使用している場合、トランスポート・ドライバで発生したシステム・エラーと関数エラーは JDE エラー・ログに書き込まれます。ドライバはエラー・メッセージを書き込み、該当する場合はエラー・コードを追加します。

Z イベントを生成するには、次のタスクを完了します。

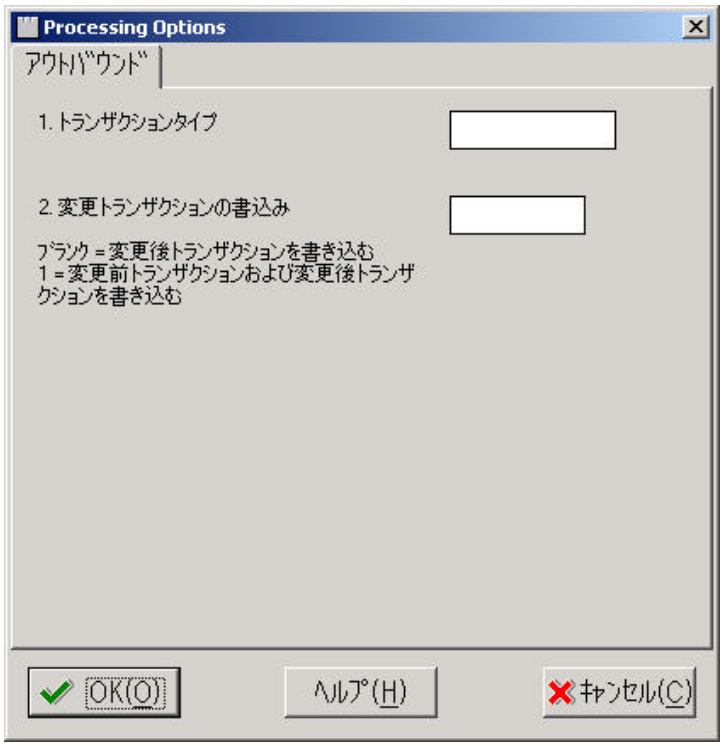
- Z イベント処理を有効にします。
- フラット・ファイル相互参照テーブル(F47002)を設定します。
- データ・エクスポート制御テーブル(F0047)を設定します。
- 処理ログ・テーブル(F0046)を設定します。
- サブシステム・ジョブが実行中であることを確認します。
- インターフェイス・テーブルからデータを除去します。
- Z イベントを定義します(F90701)。
- jde.ini ファイルを構成します。

Z イベント処理の有効化

トランザクション発生時に、マスター・ビジネス関数(MBF)によるインターフェイス・テーブルおよびサブシステム・ジョブ・マスター(F986113)へのトランザクション情報の書込みを有効にするか無効にするかを指定できます。インタオペラビリティ・トランザクションの作成機能を持つすべての送信マスター・ビジネス関数には、トランザクションの作成方法を制御する処理オプションがあります。最初の処理オプションは、インタオペラビリティ・トランザクションに対するトランザクション・タイプです。この処理オプションがブランクのままであると、送信インタオペラビリティ処理は実行されません。2 番目の処理オプションは、変更トランザクションに対する変更前トランザクションが書き込まれるかどうかを制御します。この処理オプションが 1 にセットされると、変更前トランザクションおよび変更後トランザクションがインターフェイス・テーブルに書き込まれます。この処理オプションが設定されていない場合は、変更後トランザクションのみがインターフェイス・テーブルに書き込まれます。

► Z イベント処理を有効にするには

1. トランザクションのマスター・ビジネス関数用の処理オプションが入っているアプリケーションを右クリックします。
2. メニューから[処理オプション]を選択します。



3. [送信]}タブまたは[インタオペラビリティ]タブのいずれかをクリックします。
4. トランザクション変更のために変更前および変更後トランザクションをインターフェイス・テーブルに書き込みたい場合は、トランザクション・タイプを入力して、変更前トランザクション・フラグをセットします。それから[OK]をクリックします。

参照

- インタオペラビリティ用の処理オプションを持つアプリケーションのリストは、『インタオペラビリティ』ガイドの「インタオペラビリティ・インターフェイス・テーブル情報」

フラット・ファイル相互参照の設定

Z イベントを有効化するときに、フラット・ファイル相互参照テーブル(F47002)も更新します。処理オプションに入力したトランザクション・タイプがフラット・ファイル相互参照テーブル(F47002)にマップされ、情報の取込みに使用するインターフェイス・テーブルが確定されます。このテーブルの更新には、〈フラット・ファイル相互参照〉プログラム(P47002)を使用します。

データ・エクスポート制御の設定

送信データの生成は、データ・エクスポート制御テーブル(F0047)を介して制御されます。このテーブルの更新には、〈データ・エクスポート制御〉プログラム(P0047)を使用します。トランザクション・タイプとオーダー・タイプごとに送信データを処理する Z イベント・ジェネレータを指定する必要があります。特定のトランザクション・タイプを複数のサードパーティ・アプリケーションに送信するには、データ・エクスポート制御テーブル(F0047)内で送信先ごとに個別のエントリを作成し、そのトランザクション・タイプを個々の送信先に関連付けます。各トランザクションの発生時に呼び出されるサードパーティ関数の名前を個別に指定することをお勧めします。トランザクションを通知するためのデータがすべて提供され、トランザクションを取り込むことができるように、キー値が提供されます。

▶ データ・エクスポート制御を設定するには

イベント生成をサポートしているアプリケーションから、〈データ・エクスポート制御〉アプリケーションを選びます。その他に、[略式コマンド]行に P0047 と入力して、〈データ・エクスポート制御の処理〉フォームにアクセスする方法もあります。

1. 〈データ・エクスポート制御 s〉で、[追加]をクリックします。
2. 〈データ・エクスポート制御の改訂〉で、次のフィールドに値を入力します。
   - トランザクション
   - オーダータイプ
3. 詳細ローごとに、次の項目を入力します。
   - 関数名

     Windows NT:"_CallOnUpdate@36"

     UNIX:"CallOnUpdate"

     AS/400:"CallOnUpdate"
   - Function Library

     Windows NT:"ERP Bin32 Path¥zevg.dll"

     UNIX(HP):"ERP Bin32 Path¥libzevg.sl"

     UNIX(AIX、SUN):"ERP Bin32 Path¥libzevg.so"

     AS/400:"ERP Bin32 Path¥ZEVG"
   - 追加/挿入のイベントを生成するには、[追加用実行]カラムに"1"を入力します。更新、削除、および照会についても同様の手順を実行します。
   - 送信サブシステム・バッチ処理からオブジェクトを起動するには、[即時開始]カラムに 1 を入力します。

     このカラムは、送信スケジューラ・バッチ処理に影響を与えません。

[順序]フィールドは行ごとに自動的に増分されます。

処理ログ・テーブル

Z イベント・ジェネレータは、処理ログ・テーブル(F0046)を使用します。このテーブルには、インタオペラビリティ・トランザクションのキーと正常に処理されたカラムが含まれています。順序番号、トランザクション・タイプ、オーダー・タイプ、関数名、および関数ライブラリは、データ・エクスポート制御テーブル(F0047)から取得されます。〈インタオペラビリティ汎用送信サブシステム UBE〉レポート(R00460)または〈インタオペラビリティ汎用送信スケジューラ UBE〉レポート(R00461)で処理されたトランザクションごとに、処理ログ・テーブル(F0046)にベンダ固有のレコードが逐次作成されます。たとえば、データ・エクスポート制御テーブル(F0047)を使用して 3 つのベンダがトランザクションの準備をした場合、トランザクションごとに 1 つずつ、合計 3 つのレコードが処理ログ・テーブル(F0046)に作成されます。ベンダ固有のオブジェクトがトランザクションを正常に処理した場合、処理ログ・レコードが更新され、正常に処理したカラムに Y がセットされます。ベンダ固有のオブジェクトがインタオペラビリティ・トランザクションを正常に処理したかどうかを判断するには、〈処理ログ〉プログラム(P0046)を使用します。

インターフェイス・テーブルを除去する UBE は、処理ログ・テーブル(F0046)内の情報に基づいて実行されます。

このデータは変更できません。

サブシステム・ジョブが実行中であることの確認

アプリケーションのマスター・ビジネスがサブシステム・ジョブ・マスター(F986113)にレコードを追加すると、サブシステム・ジョブが開始されます。サブシステム・ジョブとは、サブシステム・ジョブ・マスター(F986113)からのレコードを処理する継続的なジョブです。サブシステム・ジョブが実行中であることを確認する必要があります。

注意:

レコードの処理が完了すると、サブシステム・ジョブは終了する代わりに、データ待ち行列内の新規データをチェックします。サブシステム・ジョブは、終了するまで実行されています。

サブシステム・ジョブをスケジュールできます。

参照

『システム・アドミニストレーション』ガイドの次のトピックを参照してください。

- サブシステム・ジョブが実行中かどうかを確認する方法については、「ERP 9.0 サブシステムのジョブ・レコードの検討」
- サブシステム・ジョブの停止と終了については「ERP 9.0 サブシステムの終了」
- ジョブのスケジュールリングについては「Scheduler アプリケーション」

インターフェイス・テーブルのデータ除去

Z イベントを受信した後、インターフェイス・テーブルからデータを除去する必要があります。処理ログ・テーブル(F0046)に除去 UBE を入力して、インターフェイス・テーブルを除去できます。

参照

『インタオペラビリティ』ガイドの次のトピックを参照してください。

- 除去バッチ処理を持つアプリケーションのリストについては、「インタオペラビリティ・インターフェイス・テーブル情報」
- インターフェイス・テーブルからデータを除去する方法については「インターフェイス・テーブル情報の除去」

Z イベントの定義

Z イベントを定義するには、〈インタオペラビリティ・イベント記述〉プログラム(P90701)を使用します。Z イベントを定義した後は、必ず状況を変更してイベントをアクティブ化してください。イベントがインタオペラビリティ・イベント記述テーブル(F90701)に定義されていないと、システム呼出しでエラー・メッセージが戻されます。

注:

イベントの生成時に、次のメッセージが表示されることがあります。

RDEL0000045 – Could not open the tables for reliable event delivery (F90703 and F90704).(高信頼イベント配信のテーブルがオープンできませんでした(F90703 および F90704)。)Reliable event delivery will be disabled.(高信頼イベント配信は無効になります。)

これは単なる警告メッセージです。高信頼イベント配信機能を使用していない場合は無視してください。

参照

- イベントの定義方法と、イベントが見つからない場合のエラーの説明については、『インタオペラビリティ』ガイドの「イベントの定義」

Z イベントの順序設定

Z イベントを定義するときに、イベントが高信頼型であるか揮発型であるかを指定します。イベントを揮発型として定義すると、システムでは自動的にイベント順序が設定され、各イベントが正しい順序で配信されることが保証されます。揮発型イベントには、J.D. Edwards 自動採番機能を使用してスタンプが設定されます。

Z イベントの順序設定の場合、Z イベント・ジェネレータ ZEVG は、Z イベント順序設定バケットから次の番号を取得して、順序設定処理のために EVN カーネルに送ります。J.D. Edwards で保証しているのは、特定のタイプのイベント・ジェネレータに対する順序のみであることに注意してください。これは、Z イベント処理に伴う本来の遅延によるもので、先に発生したイベントに後の順序番号が設定される可能性があります。

イベント順序設定はパフォーマンスに影響します。イベント順序設定はオフにできます。また、タイムアウト値を定義し、システムに対してイベントが順不同の場合に欠落イベントのチェックを停止するように指示することもできます。フラグとタイムアウトの設定は、jde.ini ファイルの[INTEROPERABILITY]セクションで行います。

Z イベントの jde.ini ファイルの構成

Z イベントを生成するには、エンタープライズ・サーバーの jde.ini ファイルの次のセクションを構成する必要があります。

- [JDENET_KERNEL_DEF19]
- [JDEITDRV]
- [JDENET]
- [INTEROPERABILITY]

EVN カーネルの[JDEITDRV]、および[JDENET]の設定の定義については、『インタオペラビリティ』ガイドの「イベントの jde.ini ファイルの構成」を参照してください。[INTEROPERABILITY]の設定は、次のとおりです。

```
[INTEROPERABILITY]
SequenceTimeOut=XX
XMLElementSkipNullOrZero=X
```

SequenceTimeOut 設定は揮発型イベントの順序設定用で、値は秒単位です。

デフォルトの NULL 文字列と 0(ゼロ)を設定すると、Z イベントから切り捨てられます。
XMLElementSkipNullOrZero 設定に値 0(ゼロ)を入力すると、この機能をオフにできます。

Z ファイル・イベントの XML ドキュメント・フォーマット

Z イベントの XML ドキュメントは、J.D. Edwards XML 応答フォーマットを使用します。Z イベントの XML ドキュメントの例については、付録の「XML フォーマット例(Z イベント)」を参照してください。異なるイベントでは、テーブル名やカラム名が異なっていてもかまいません。

ベンダ固有の送信機能

ベンダ固有の送信機能の目的は、ERP の送信インターフェイス・テーブル内のレコードのキー・フィールドをサードパーティに渡すことです。これらのキーを使用すると、データベース・レコードからのデータをサードパーティ・システムに合わせて処理できます。汎用の送信サブシステム・バッチ処理は、関数を呼び出します。

各ベンダ固有の関数は、処理されるトランザクションに特有のものです。データベース・レコード・データでこの関数が実際に実行するものを判別する必要があります。関数はユーザーのスペックに書き込まれ、ほとんど ERP の外部に書き込まれますが、これらの関数では J.D. Edwards により定義された、次の必須データ構造体を使用する必要があります。

データ項目	必須	I/O	説明
szUserId	Y	I	ユーザーID - 11 文字
szBatchNumber	Y	I	バッチ番号- 16 文字
szTransactionNumber	Y	I	トランザクション番号- 23 文字
mnLineNumber	Y	I	行番号- 2 文字
szTransactionType	Y	I	トランザクション・タイプ- 9 文字
szDocumentType	Y	I	伝票タイプ- 3 文字
mnSequenceNumber	Y	I	順序番号- 2 文字

リアルタイム・イベント

リアルタイム・イベントは、ERP でビジネス・トランザクションが発生したことを示す通知です。HTML、WIN32、およびターミナル・サーバーなど、任意の ERP インターフェイスを使用して、エンタープライズ・サーバー上でリアルタイム・イベントを生成できます。リアルタイム・イベントは、同期処理と非同期処理のいずれにも使用可能です。

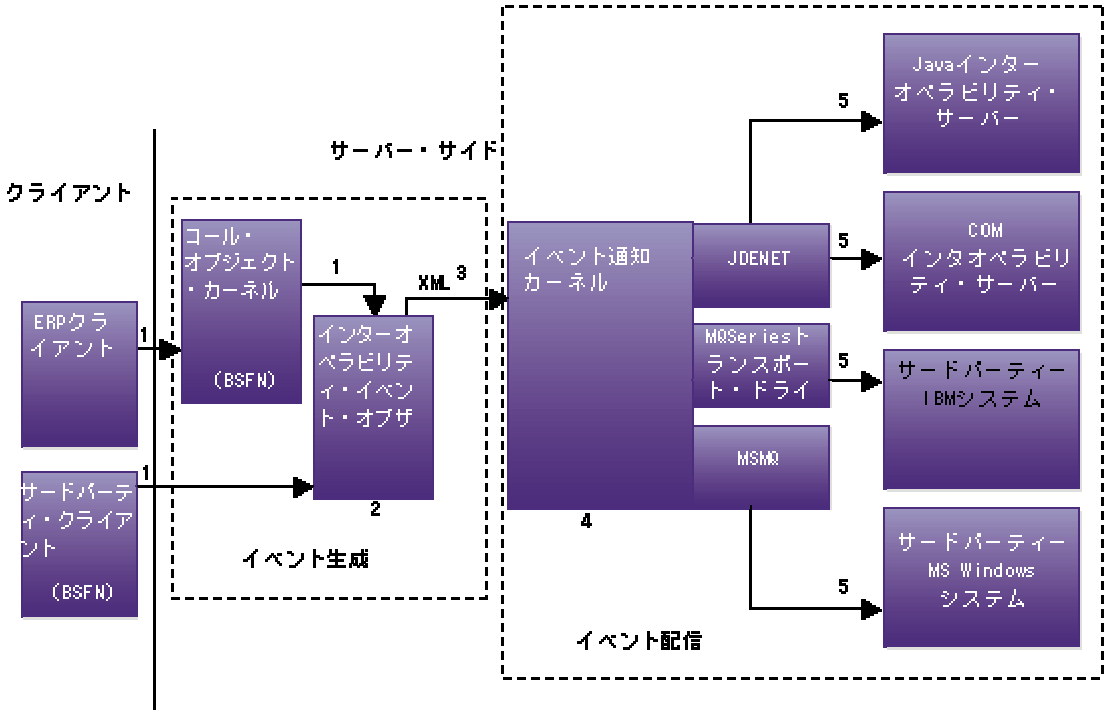
たとえば、ERP をバックエンドとして使用しているオークション・サイトは、リアルタイム・イベントを使用してデータベースを直ちに更新できます。ユーザーがオークションに新しい品目を入力すると、それが ERP システムへのトランザクションのトリガーとなります。トランザクションは直ちにキャプチャされます。Java コネクタなどのインタオペラビリティ・サーバーに通知が送られると、そこから Web エンジンにその情報が伝達されます。これにより、すべてのオークション・ユーザーが新しい品目を見ることができるように html ページが更新されます。この例は、同期処理の例を示すものです。

リアルタイム・イベント生成は、非同期処理でも利用可能です。たとえば、あるオンライン・ショップが別の仕入先にオーダーを送り(ビジネス間：B2B)、トランザクションをキャプチャすると、オーダーが仕入先のシステムに入力されます。あるユーザーが本を買ったとします。仕入先は本の出版社に購買オーダーを出し、本を受け取って配達するように運送会社に通知することになります。本のオーダー自体は ERP との購買オーダー・トランザクションとして完了できますが、運送会社の処理のため、出荷要求では一般的に採用されているフォーマットにデータを格納する必要があります。

リアルタイム・イベントは、ビジネス関数から ERP データ構造体の形でデータを受け取るシステム・コールを使用します。イベントには次のような種類があります。

単一イベント	部分的なイベントを 1 つ含みます。レシーバがシステム・コール実行時のイベント生成を必要とする場合に役立ちます。異なるイベント・タイプと共に使用することもできます。
集合体イベント	複数の部分的なイベントを含みます。レシーバが複数のイベントを含むドキュメントを必要とする場合に役立ちます。たとえばサプライ・チェーン・ソリューションでは、受注オーダー全体を、複数の部分的なイベントを含む 1 つのイベントとして供給する必要があります。
複合イベント	単一イベントのみを含みます。顧客が複数のレシーバを持ち、一部が単一イベントを要求し、別のレシーバが集合体イベントに類似した完全なイベントを要求する場合に有効です。

クライアントからのリアルタイム・イベント生成では、クライアント・ビジネス関数を用いて API を呼び出し、サーバーであるインタオペラビリティ・イベント・オブザーバ（カーネル）とインターフェイスします。サーバー側のリアルタイム・イベント生成には、リアルタイム・イベントをトリガーするイベント・オブザーバ・インターフェイス（システム API のセット）が用いられます。また、インタオペラビリティ・イベント・オブザーバ（カーネル）を使用します。イベント・オブザーバ・カーネルは、トリガーされたリアルタイム・イベントの XML ドキュメントを生成し、それらをイベント配布コンポーネントに送ります。イベント配布コンポーネントは、XML ドキュメント通知をサブスクライバに送信するときに用いられるコンポーネントと同じです。次の図にリアルタイム・イベント生成用のプロセスとデータのロジックを示し、続いて説明します。



1. ERP ビジネス関数によってイベント生成がトリガーされます。ERP の〈オブジェクト構成マネージャ〉(OCM)を使用して、ビジネス関数をエンタープライズ・サーバー上またはローカルで実行するようにマップします。

   - J.D. Edwards ビジネス関数をエンタープライズ・サーバー上で実行するようにマップした場合は、CallObject カーネル内でインタオペラビリティ・イベント・インターフェイスが呼び出されます。カーネルが、情報を Interoperability Event Observation(IEO)に送ります。
   - クライアントからリアルタイム・イベントが生成されると、クライアントのビジネス関数が該当する API を呼び出します。API は OCM を参照して、IEO カーネルの位置を判別し、データの妥当性チェック、フィルタ、およびフォーマットを実行してから、IEO カーネルに送ります。

2. IEO カーネルはリアルタイム・イベントを作成し、その確定時に XML ドキュメントを生成します。
3. IEO カーネルは XML ドキュメントをパッケージして、イベント通知(EVN)カーネルに渡します。
4. EVN カーネルは、どのトランスポート・ドライバでイベントを処理するかを確定します。
5. トランスポート・ドライバがイベントを配布します。

注:

MQSeries または MSMQ トランスポート・ドライバを使用している場合、トランスポート・ドライバで発生したシステム・エラーと関数エラーは JDE エラー・ログに書き込まれます。ドライバはエラー・メッセージを書き込み、該当する場合はエラー・コードを追加します。

イベントの固有 ID

それぞれのリアルタイム･イベントには、ERP セッション ID を含む固有 ID が付けられています。

実行記録

JDEDEBUG ログ･ファイルのトレース機能を用いて、リアルタイム･イベントの動作が記録されます。実行記録は、jde.ini ファイルで有効または無効に切り替えることができます。実行記録は次の 2 つの時点で発生します。

- 受け取ったパラメータおよび呼び出された API を、システム･コールがログに記録します。
- インタオペラビリティ・イベント･オブザーバ処理の間、追加のデバッグ情報をカーネルがログに記録します。ログへの記録は、[INTEROPERABILITY]セクションで LEVEL キーによって制御します。次の表に、LEVEL キーで使用可能な値をいくつか示します。

[INTEROPERABILITY]

LEVEL=		指定されたインタオペラビリティ・イベント・データをデバッグ・ログ･ファイルに書き込みます。1 つまたは複数の許容ログ設定を指定できます。複数の値はカンマで区切ってください。次に例を示します。 [INTEROPERABILITY] LEVEL=EVENTS,DATA **注:** どのようなログ記録操作においても、これらの設定のいずれかを有効にすると、性能に影響を与え、多量のデータが書き込まれる場合があります。
	EVENTS	IEO カーネルのイベントのフローを記録するには、この値を使用します。イベント･データの受信およびイベントの送信が記録されますが、イベント･データの値は記録されません。これがデフォルトのレベルです。LEVEL キーがない場合は、LEVEL=EVENTS と設定した場合と同じになります。
	DATA	IEO カーネルのイベント･データの値とイベントのフローを記録します。このレベルには、EVENTS スイッチでログに記録するすべてのデータが含まれます。
	PERF	受け取ったイベントの数と処理にかかった時間に関する統計を記録します。
	DOC	送信 XML ドキュメントは、ディスク上のテンポラリ・ファイルに書き込まれます。デバッグ･ログを有効にした場合、ドキュメントの位置がデバッグ・ログに書き込まれます。ドキュメントの位置は、次に示すうちのいずれかになります。 • [Interoperability]セクションのキーTempFileDir に値が設定されていれば、ファイルはその位置に書き込まれます。次に例を示します。 [INTEROPERABILITY] TempFileDir=C:¥XML_DOCUMENTS • キーTempFileDir に値が設定されていない場合、JDE ログとデバッグ･ログが書き込まれるのと同じディレクトリに、ファイルが書き込まれます。 **注意:** LEVEL=DOC キーを設定すると、すべてのリアルタイム・イベントがディスクに書き込まれ、エンタープライズ・サーバーのパフォーマンスが大幅に低下する可能性があります。本稼働用環境や、QA 環境でのストレス・テストには、LEVEL=DOC 設定を使用しないことをお勧めします。

	TRACE	このスイッチは、IEO カーネルの実行をトレースし、データをデバッグ･ログに書き込みます。記録されるデータの量が多いため、このレベルはデバッグを行う場合のみ使用してください。

注:

デバッグ・ログが有効化されているかどうかは、LEVEL=DOC 設定に影響しません。LEVEL キーの他のすべての値（TRACE など）は、デバッグ・ログの有効または無効設定の影響を受けます。

jde.ini ファイルの[INTEROPERABILITY]セクションで SaveEVNDoc キーを設定して、EVN ドキュメントを記録することもできます。SaveEVNDoc は LEVEL=DOC に似ていますが、IEO カーネルではなく EVN カーネルに適用されます。SaveEVNDoc のデフォルト値はゼロ(0)で、ENV ドキュメントは保存されません。EVN ドキュメントを保存するには、値を 1 に変更します。EVN ドキュメントは、他のディレクトリを指定しなければ、JDE ログおよびデバッグ・ログと同じディレクトリに保存されます。TempFileDir を使用すると、次のようにディレクトリを指定できます。

```
[INTEROPERABILITY]
    SaveEVNDoc=1
    TempFileDir=C:¥XML_Documents
```

参照

- jde.ini ファイル内のデバッグ設定については『システム・アドミニストレーション』ガイドの「jde.ini File（jde.ini ファイル）」
- デバッグ・ログの有効化と無効化については、『開発ツール』ガイドの「デバッグ・トレーシング」

リアルタイム・イベント API

システム API は、システム･コールがサーバーから呼び出されたかクライアントから呼び出されたかを判断できます。リアルタイム･イベントの生成には次に挙げる API が利用可能です。

- jdeIEO_EventInit()
- jdeIEO_EventAdd()
- jdeIEO_EventFinalize()
- jdeIEO_CreateSingleEvent()
- jdeIEO_IsEventTypeEnabled()

注:

これらのイベントについては、オンライン APIドキュメンテーションを参照してください。

例:インタオペラビリティ・イベントのインターフェイス・コール

次の例では、単一イベントの作成方法を示します。

1. リアルタイム・イベントに与える値を決定するデータ構造体を設計します。

```
typedef struct tagDSD55RTTEST
  {
  char              szOrderCo[6];
  char              szBusinessUnit[13];
  char              szOrderType[3];
  MATH_NUMERIC      mnOrderNo;
  MATH_NUMERIC      mnLineNo;
  JDEDATE           jdRequestedDate;
  char              szItemNo[27];
  char              szDescription1[31];
  MATH_NUMERIC      mnQtyOrdered;
  MATH_NUMERIC      mnUnitPrice;
  MATH_NUMERIC      mnUnitCost;
  char              szUserID[11];
  } DSD55RTTEST, *LPDSD55RTTEST;
```

2. ビジネス関数ヘッダー・ファイル内でデータ構造体オブジェクトを定義します。
3. ビジネス関数ソースを、jdeIEO_CreateSingleEvent を呼び出すように修正します。

```
JDEBFRTN(ID) JDEBFWINAPI RealTimeEventsTest (LPBHVRCOM  lpBhvrCom,
   LPVOID  lpVoid,
   LPDSD55REALTIME lpDS)
{
/* Define Data Structure Object */
DSD55RTTEST     zRTTest     = {0};
  IEO_EVENT_RETURN    eEventReturn    = eEventCallSuccess;
IEO_EVENT_ID     szEventID     = {0};

0Populate required members

/* Now call the API */
  szEventID = jdeIEO_CreateSingleEvent ( lpBhvrCom,
       "RealTimeEventsTest",
        "JDERTOUT",
        "SalesOrder",
        "D55RTTEST",
       &zRTTest,
       sizeof(zRTTest),
       0,
       &eEventReturn    );

  /* Error in jdeFeedCallObjectEvent is not a critical error
          and should only be treated as a warning */
     if( eEventReturn != eEventCallSuccess )
  {
     /* LOG the Warning and return */
     return ER_WARNING;
```

次の例では、集合体イベントの作成方法を示します。

```
IEO_EVENT_RETURN eEventReturn = eEventCallSuccess;
IEO_EVENT_ID szEventID = jdeIEO_EventInit (pBhvrCom,
eEventAggregate,
"MyFunction1",
"JDESOOUT" // EventType for AggregateEvent
"EventScope1",
0,
&eEventReturn);
0

eEventReturn = jdeIEO_EventAdd (pBhvrCom,
szEventID,
"MyFunction2",
NULL,
"D55TEST01",
&zD55TEST01,
sizeof(zD55TEST01));
0

eEventReturn = jdeIEO_EventAdd (pBhvrCom,
szEventID,
"MyFunction3",
NULL,
"D55TEST02",
&zD55TEST02,
sizeof(zD55TEST02));
0

eEventReturn = jdeIEO_EventAdd (pBhvrCom,
EventID,
"MyFunction3",
NULL,
"D55TEST03",
&zD55TEST03,
sizeof(zD55TEST03));
0

eEventReturn = jdeIEO_EventFinalize (pBhvrCom,
EventID,
"MyFunction4");
```

次の例では、複合イベントの作成方法を示します。

```
IEO_EVENT_RETURN eEventReturn = eEventCallSuccess;
IEO_EVENT_ID szEventID = jdeIEO_EventInit (pBhvrCom,
eEventComposite,
 □MyFunction1",
 □JDESOOUT" // EventType for CompositeEvent
 □EventScope1",
0,
&eEventReturn );
0

eEventReturn = jdeIEO_EventAdd ( pBhvrCom,
             szEventID,
             □MyFunction2",
             □SODOCBEGIN", // EventType for SingleEvent
             □D55TEST01",
             &zD55TEST01,
             sizeof(zD55TEST01));
0

eEventReturn = jdeIEO_EventAdd ( pBhvrCom,
            szEventID,
            □MyFunction3",
            □SOITEMADD",  //EventType for SingleEvent
            □EventScope3",
            □D55TEST02",
            &zD55TEST02,
            sizeof(zD55TEST02));
0

eEventReturn  = jdeIEO_EventFinalize (pBhvrCom,
          EventID,
          □MyFunction4");
```

システム・コールから返されたエラーは、ビジネス・プロセスを停止させるほど重大なものでない場合があります。重大でないエラーは警告としてフラグが立てられ、ログ・ファイルに記録されます。ビジネス関数がサーバー側にあれば、警告は callobject カーネル・ログに記録されます。ビジネス関数がクライアント側にある場合、警告はクライアント・ログ・ファイルに記録されます。

次の例は、複合リアルタイム・イベントが記述された XML ファイルを示します。このイベントは、12/31/2000 の正午ごろに届いたビジネス関数 F4211FSEditLine への呼出しと、12:00:01.000 に生成されたリアルタイム・イベントで構成されています。

```
<?xml version='1.0' encoding='utf-8' ?>
<jdeResponse type='realTimeEvent' user='JDE1214225' ses
sion='1234.786321234' environment='XDEVNIS2'>
  <header>
    <eventVersion>1.0</eventVersion>
    <type>JDESOOUT</type>
    <scope>SalesOrder</scope>
    <user>JDE1214225</user>
    <application>P0101</application>
    <version>XJDE101</version>
    <sessionID>1234.786321234</sessionID>
    <environment>XDEVNIS2</environment>
    <host>HP9000B</host>
    <eventID>
      HP9000B-1234-1231200012000100-JDE1214225-FFA123ECBBA123EC
    </eventID>
    <date>12312000</date>
    <time>120001000</time>
  </header>

 body elementCount='1'>
    <PartialEvent name='F4211FSEditLine' type='SOEDTLINE' executionOrd
er='1' parameterCount='12'>
  <szOrderCo type='String'>JD Edwards</szOrderCo>
      <szBusinessUnit type='String'>Mountain Region</szBusinessUnit>
      <szOrderType type='String'>SO</szOrderType>
   <mnOrderNo type='MATH_NUMERIC'>13209847</mnOrderNo>
      <mnLineNo type='MATH_NUMERIC'>122</mnLineNo>
      <jdRequestedDate type='Date'>12312000</jdRequestedDate>
      <szItemNo type='String'>12243234</szItemNo>
      <szDescription type='String'>Bicycle</szDescription>
      <mnQtyOrdered type='MATH_NUMERIC'>1</mnQtyOrdered>
      <mnUnitPrice type='MATH_NUMERIC'>249.99</mnUnitPrice>
      <mnUnitCost type='MATH_NUMERIC'>213.23</mnUnitCost>
<szUserID type='String'>JDE1214225</szUserID>
    </PartialEvent>
  </body>
</jdeResponse>
```

リアルタイム・イベントの生成

次のリストに、リアルタイム・イベントの生成タスクを示します。

- OCM を設定します。
- リアルタイム・イベントを定義します。
- jde.ini ファイルを構成します。

注:

イベントの生成時に、次のメッセージが表示されることがあります。

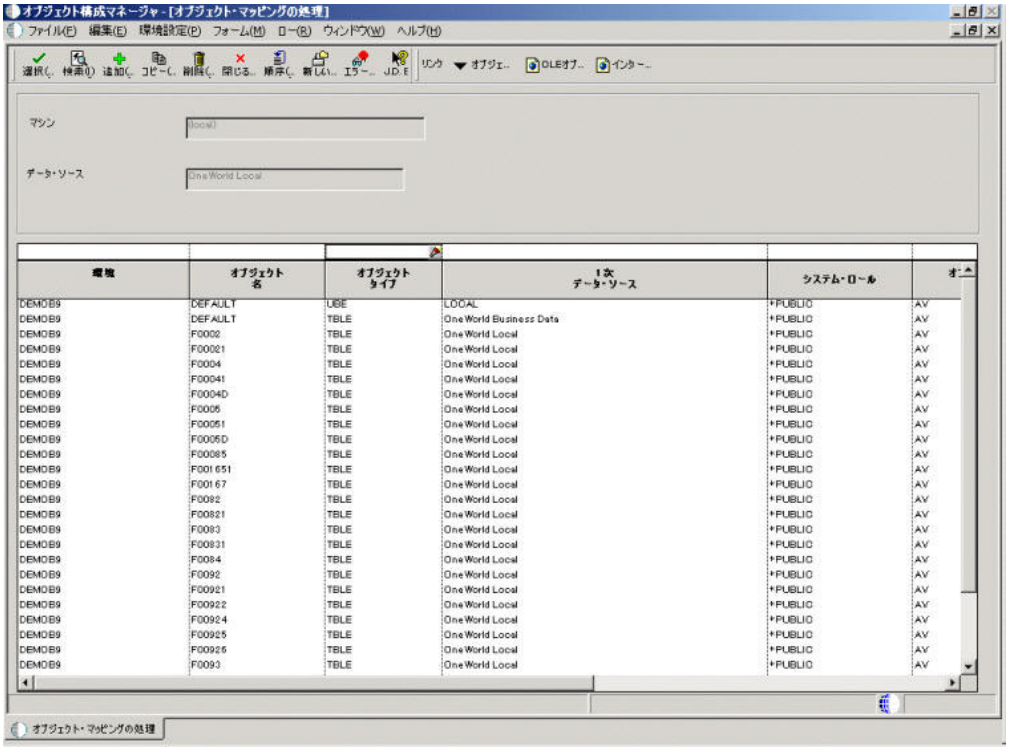
> RDEL0000045 – Could not open the tables for reliable event delivery (F90703 and F90704). (高信頼イベント配信のテーブルがオープンできませんでした(F90703 および F90704)。) Reliable event delivery will be disabled.(高信頼イベント配信は無効になります。)

これは単なる警告メッセージです。高信頼イベント配信機能を使用していない場合は無視してください。

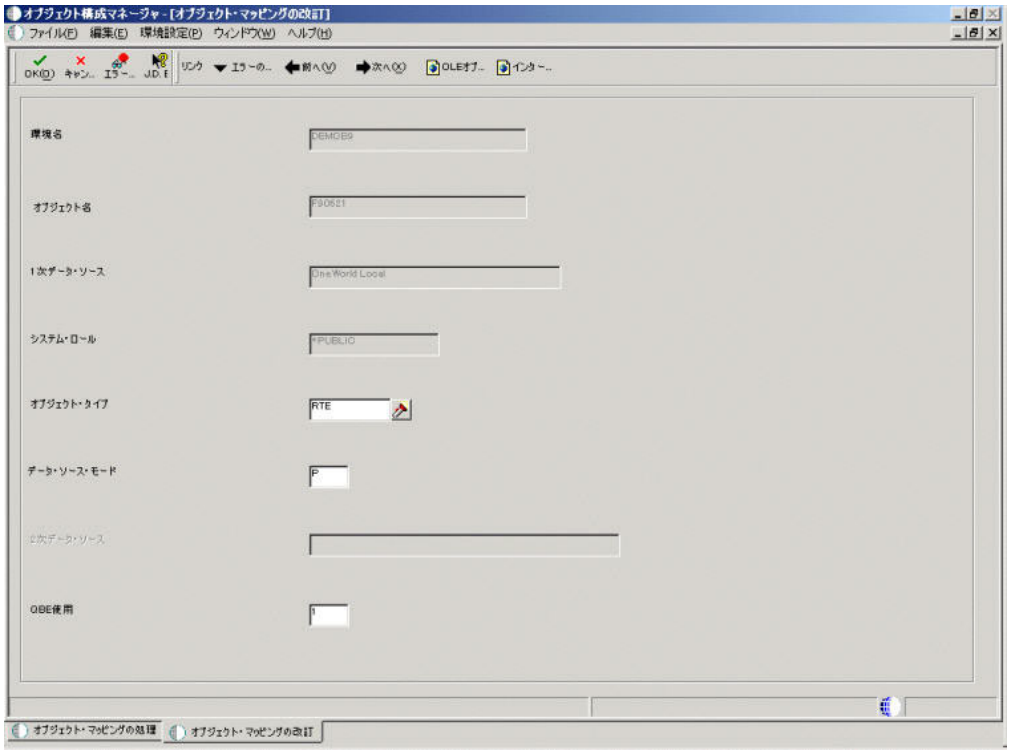
リアルタイム・イベント用の OCM の設定

〈オブジェクト構成マネージャ〉(OCM)を、システム呼出しで IEO カーネルを検索できるように構成します。ビジネス関数がクライアントにマップされている場合は、IEO カーネルが見つからないと、システム呼出しからクライアントにエラーが戻されます。ビジネス関数がサーバーにマップされている場合、エラーは Callobject カーネルの jde.log に記録されます。

OCM を構成するときには、特定の環境を指定してください。また、2 つの重複したマッピングが同時にアクティブ状態にならないようにしてください。次の例で、環境と状況を示します。



OCM を構成するには、〈オブジェクト・マッピングの改訂〉フォームにアクセスして、次に示すように[オブジェクト・タイプ]フィールドに”RTE”と入力します。



クライアント上で OCM マッピングが正しく構成されていない場合、次のメッセージが jde.log に書き込まれ、イベントは生成されません。

RT0000011 jdeIEO_EventInit:Unable to find the server（サーバーが見つかりません。）

サーバー上で OCM マッピングが正しく構成されていない場合、エラー・メッセージは生成されません。システム・コールは、IEO カーネルの位置としてローカル・サーバーのデフォルト値を使用します。

OCM で構成されたマシン上に IEO カーネルが見つからない場合、次のようなエラーが発生することがあります。

RT0000004 jdeIEO_EventInit:ReceiveMsg failed.（ReceiveMsg は失敗しました。）Error = <error test>

参照

『CNC インプリメンテーション』ガイドの「オフライン処理」を参照してください。

- 〈オブジェクト構成マネージャ〉ツールについては「オブジェクト構成マネージャの処理」
- イベント処理用にビジネス関数をマップする方法については「オブジェクトのマッピング」

リアルタイム・イベントの定義

リアルタイム・イベントを定義するには、〈インタオペラビリティ・イベントの定義〉プログラム(P90701)を使用します。リアルタイム・イベントを定義した後は、必ず状況を active（アクティブ）に変更してイベントをアクティブ化してください。イベントがインタオペラビリティ・イベント記述テーブル(F90701)に定義されていないと、システム呼出しでエラー・メッセージが戻されます。

参照

- イベントの定義方法と、イベントが見つからない場合のエラーの説明については、『インタオペラビリティ』ガイドの「イベントの定義」

イベントの順序設定

リアルタイム・イベントを定義するときに、イベントが高信頼型であるか揮発型であるかを指定します。イベントを揮発型として定義すると、システムでは自動的にイベント順序が設定され、各イベントが正しい順序で配信されることが保証されます。揮発型イベントには、J.D. Edwards 自動採番機能を使用してスタンプが設定されます。

リアルタイム・イベントの順序設定の場合、システム呼出し jdeIEO_EventFinalize は、リアルタイム・イベント順序設定バケットからイベント番号を取得して IEO カーネルに送ります。IEO カーネルは、生成するイベントの一部としてイベント番号を組み込みます。その後、イベントを EVN カーネルに送ります。EVN カーネルは最後に送られたイベントを記憶し、受け取ったイベントに含まれていたイベント番号に基づいて順序設定を行います。

イベント順序設定はパフォーマンスに影響します。イベント順序設定はオフにできます。また、タイムアウト値を定義し、システムに対してイベントが順不同の場合に欠落イベントのチェックを停止するように指示することもできます。フラグとタイムアウトの設定は、jde.ini ファイルの[INTEROPERABILITY]セクションで行います。

リアルタイム・イベントの jde.ini の構成

リアルタイム・イベントを生成するには、エンタープライズ・サーバーの jde.ini ファイルの次のセクションを構成する必要があります。

- [JDENET_KERNEL_DEF19]
- [JDENET_KERNEL_DEF20]
- [JDEITDRV]
- [JDENET]
- [INTEROPERABILITY]

注:

カーネルの[JDEITDRV]および[JDENET]の設定の定義については、『インタオペラビリティ』ガイドの「イベントの jde.ini ファイルの構成」を参照してください。

[INTEROPERABILITY]の設定は、次のとおりです。

```
[INTEROPERABILITY]
    SequenceTimeOut=XX
    XMLElementSkipNullOrZero=X
```

SequenceTimeOut 設定は揮発型イベントの順序設定用で、デフォルト値は 10 秒です。

デフォルトの NULL 文字列と 0(ゼロ)を設定すると、リアルタイム・イベントから切り捨てられます。XMLElementSkipNullOrZero 設定に値 0(ゼロ)を入力すると、この機能をオフにできます。

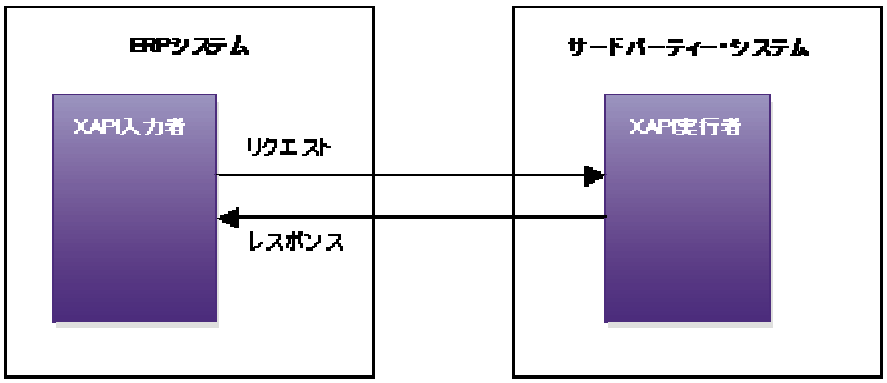
XAPI

XAPI は ERP サービスであり、トランザクションの発生時に ERP トランザクションをキャプチャし、サードパーティ・ソフトウェア、エンドユーザー、および他の J.D. Edwards システムを呼び出してリターン応答を取得します。XAPI はリアルタイム・イベントに類似しており、リアルタイム・イベントと同じインフラストラクチャを使用します。リアルタイム・イベントと XAPI イベントの違いは、XAPI イベントのサブスクライバが入力者に返信を戻すことです。XAPI イベントには、ユニークな XAPI イベント名および戻り時に呼び出されるビジネス関数などの構造化データのセットが含まれています。リアルタイム・イベントと同様に、XAPI イベントは HTML、WIN32、およびターミナル・サーバーなど、任意の ERP インターフェイスを使用して、エンタープライズ・サーバー上でリアルタイム・イベントを生成できます。

XAPI 構造は送信イベントを送信し、サードパーティ・システムから返信を受け取ります。また、2 つの ERP システム間で完全な要求/返信接続を提供します。

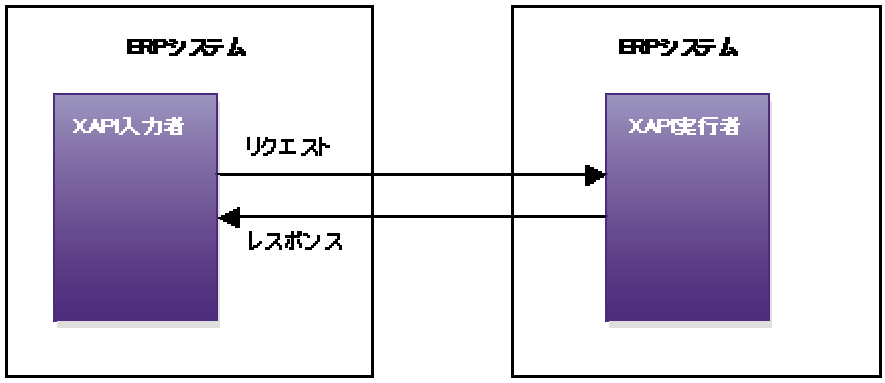
ERP システム内でイベントが生成されると、処理のためにサードパーティ・システムに送信されます。サードパーティ・システムからは、ERP システムに応答が送信されます。

次の図に、ERP からサードパーティ・システムへの XAPI 処理のロジックを示します。



- XAPI の入力者である ERP が要求を送信します。
- 要求がサードパーティ・システムに送信されます。
- XAPI の実行者であるサードパーティ・システムは、要求を処理して XAPI 入力者に応答を送信します。

XAPI は、2 つの異なる ERP システムが相互に通信するための手段も提供します。次の図に、ある ERP システムから他の ERP システムへの XAPI 処理のロジックを示します。

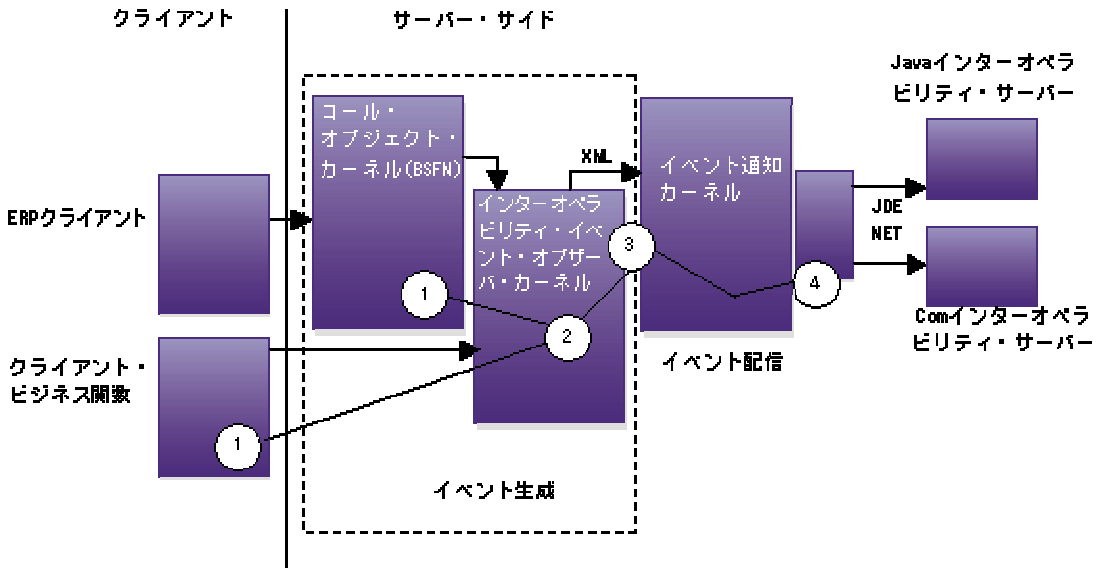


- XAPI の入力者である ERP システム 1 が要求を送信します。
- 要求は ERP システム 2 に送信されます。このシステムの環境は、ERP システム 1 と同じ共有環境であっても異なる環境であってもかまいません。
- XAPI の実行者である ERP システム 2 は、要求を処理して XAPI 入力者である ERP システム 2 に応答を送信します。
- XAPI の入力者である ERP システム 1 が応答を処理します。

XAPI 送信イベント

XAPI 構造は、XAPI 送信イベント生成をサポートしています。XAPI 送信イベントの生成プロセスは、リアルタイム・イベントの場合とまったく同じです。

次の図に、サードパーティ・システムに送信される XAPI 送信イベントのプロセス・フローを示します。



1. クライアントから XAPI イベントが生成されると、クライアントのビジネス関数が該当する API を呼び出します。この API は OCM を参照し、IEO カーネルの保管場所を判断します。API がデータの確認、フィルタ、およびフォーマットを実行します。XAPI イベントがエンタープライズ・サーバーから生成されると、J.D. Edwards ビジネス関数が CallObject カーネル内でインタオペラビリティ・イベント・インターフェイスを呼び出します。データが部分的なイベントとして IEO カーネルに送信されます。
2. IEO カーネルは XAPI イベントを作成し、その確定時に XML ドキュメントを生成します。
3. IEO カーネルは XML ドキュメントをパッケージして、EVN カーネルに渡します。
4. EVN カーネルはイベントを処理するトランスポート・ドライバを確定し、JDENet は情報をサブスクライバに配布します。

注:

現在、XAPI では MQSeries または MSMQ を使用しません。MQSeries または MSMQ トランスポート・ドライバを使用してイベントを受信するようにシステムを構成している場合は、インタオペラビリティ・イベント記述テーブル(F90701)内で定義されているイベントをすべて受信することになります。

XAPI イベント API

XAPI 呼出しの生成には次に挙げる API が利用可能です。

- jdeXAPI_Init
- jdeXAPI_Add
- jdeXAPI_Finalize
- jdeXAPI_Free
- jdeXAPI_SimpleSend
- jdeXAPI_ISCallTypeEnabled
- jdeXAPI_CALLS_ENABLED

注:

これらのイベントについては、オンライン APIドキュメンテーションを参照してください。

例:XAPI イベント作成 API の使用

次のサンプル・コードは、XAPI イベントの作成方法を示します。

```
#ifdef jdeXAPI_CALLS_ENABLED
    XAPI_CALL_ID        ulXAPICallID      = 0;
XAPI_CALL_RETURN                eXAPICallReturn   = eEventCallSuccess;
#endif

DSD4205010A     dsD4205010A                       = {0};     /*Query Header*/
DSD4205010B     dsD4205010B                       = {0};     /*Query Detail*/
```

```
#ifdef jdeXAPI_CALLS_ENABLED
if(jdeXAPI_IsCallTypeEnabled("XAPIOPOUT") &&
        jdeXAPI_IsCallTypeEnabled("XAPIOPIN") )
{
        bXAPIInUse = TRUE;
}
#endif

        /*-------------------------------------------------------*/
        /* Call XAPIInit    */

        #ifdef jdeXAPI_CALLS_ENABLED
        if(bXAPIInUse == TRUE)
        {
                ulXAPICallID = jdeXAPI_Init( lpBhvrCom,
                                                "SendOrderPromiseRequest",
                                                "XAPIOPOUT",
                                                NULL,
                                                &eXAPICallReturn);
                if (eXAPICallReturn != eEventCallSuccess)
                {
                        bExit = TRUE;
                }
        }
        #endif


        /*-------------------------------------------------------*/
        /* Adding Header Information               */

                #ifdef jdeXAPI_CALLS_ENABLED
                if(bXAPIInUse == TRUE)
                {
                        eXAPICallReturn = jdeXAPI_Add( lpBhvrCom,
                                        ulXAPICallID,
                                        "SendOrderPromiseRequest",
                                        "D4205010A",
                                        &dsD4205010A,
                                        sizeof(DSD4205010A));
                        if (eXAPICallReturn != eEventCallSuccess)
                        {
                                bExit = TRUE;
                        }
                }
                #endif

        /*-------------------------------------------------------*/
        /* Loading Detail Information             */


                        #ifdef jdeXAPI_CALLS_ENABLED
                        if(bXAPIInUse == TRUE)
                        {
                                eXAPICallReturn = jdeXAPI_Add( lpBhvrCom,
```

```
                                        ulXAPICallID,
                                        "SendOrderPromiseRequest",
                                        "D4205010B",
                                        &dsD4205010B,
                                        sizeof(DSD4205010B));
                                if (eXAPICallReturn != eEventCallSuccess)
                                {
                                        bExit = TRUE;
                                }
                        }
                        #endif

                        #ifdef jdeXAPI_CALLS_ENABLED
                        if(bXAPIInUse == TRUE)
        /*-----------------------------------------------*/
                /* Finalize             */
                        {
                                eXAPICallReturn = jdeXAPI_Finalize( lpBhvrCom,
                                        ulXAPICallID,
                                        "SendOrderPromiseRequest",
                                        "OrderPromiseCallback");
                                if (eXAPICallReturn != eEventCallSuccess)
                                {
                                        bExit = TRUE;
                                }
                        }
                        #endif

    #ifdef jdeXAPI_CALLS_ENABLED
    if (eXAPICallReturn != eEventCallSuccess)
    {
                /*-----------------------------------------------*/
                /* CleanUp */

                if(bXAPIInUse == TRUE)
                {
                        jdeXAPI_Free( lpBhvrCom,
                                        ulXAPICallID,
                                        "SendOrderPromiseRequest");
                }
    }
    #endif
```

例：XAPI 送信イベントの XML

次の例に、XAPI イベント用の XML テンプレートを示します。

```
xml version="1.0" encoding="utf-8" ?>
<jdeResponse type="realTimeEvent" user="KL5449350" session="22558100.1004460662"
subtype="XAPICall" environment="DV7333">
<event>
<header>
```

<eventVersion>1.0</eventVersion>
<type>XAPIOPOUT</type>
<user>KL5449350</user>
<application>APIDRV</application>
<version />
<sessionID>22558100.1004460662</sessionID>
<environment>DV7333</environment>
<host>DEN-PP6954083</host>
<sequenceID>DEN-PP6954083_1540_10302001095648_KL5449350_1</sequenceID>
<date>10302001</date>
<time>095649</time>
<scope />
<codepage>utf-8</codepage>
</header>
<body elementCount="3">
<detail date="10302001" name="APIDRVFunction" time="9:56:48" type="" DSTMPL="D4205010A"
executionOrder="1" parameterCount="23">
<szRequestId type="String">1234567</szRequestId>
<szUserId type="String">TestUser</szUserId>
<szQueryMode type="String">Test</szQueryMode>
<szCustomerName type="String">John Doe</szCustomerName>
<mnCustomerId type="Double">12345</mnCustomerId>
<szCustomerGroup type="String">Group 1</szCustomerGroup>
<szAddress1 type="String">Line 1</szAddress1>
<szAddress2 type="String">Suite 1</szAddress2>
<szAddress3 type="String">123 E. Main</szAddress3>
<szPostalCode type="String">50001</szPostalCode>
<szCity type="String">Centennial</szCity>
<szCounty type="String">Arap</szCounty>
<szStateProvince type="String">CO</szStateProvince>
<szCountry type="String">US</szCountry>
<szBusinessObjective type="String" />
<mnTraceDepth type="Double">0</mnTraceDepth>
<mnPenaltyCostAdjustment type="Double">0</mnPenaltyCostAdjustment>
<szOrderNumber type="String">1000</szOrderNumber>
<nAllowBackorders type="Int">49</nAllowBackorders>
<nAllowSubstitution type="Int">48</nAllowSubstitution>
<nAllowPartialLineShip type="Int">49</nAllowPartialLineShip>
<nAllowPartialOrderShip type="Int">49</nAllowPartialOrderShip>
<nAllowMultisource type="Int">49</nAllowMultisource>
</detail>
<detail date="10302001" name="APIDRVFunction" time="9:56:49" type="" DSTMPL="D4205010B"
executionOrder="2" parameterCount="17">
<mnLineNumber type="Double">1</mnLineNumber>
<mnCacheLineNumber type="Double">1</mnCacheLineNumber>
<mnItemNumber type="Double">2222</mnItemNumber>
<sz2ndItemNumber type="String">1234567</sz2ndItemNumber>
<sz3rdItemNumber type="String">2234567</sz3rdItemNumber>
<szOrderUnit type="String">123</szOrderUnit>
<mnOrderQuantity type="Double">12</mnOrderQuantity>
<szPlanningUnit type="String">ECL</szPlanningUnit>
<mnPlanningQuantity type="Double">12</mnPlanningQuantity>
<mnPlanningMultiple type="Double">1</mnPlanningMultiple>
<mnPlanningUnitPrice type="Double">1234</mnPlanningUnitPrice>
<jdRequestDate type="Date">10302001</jdRequestDate>

```
  <szShippingGroup type="String">Ship Group</szShippingGroup>
  <szMultiSource type="String">MS</szMultiSource>
  <nAllowPartialLineShip type="Int">49</nAllowPartialLineShip>
  <nAllowBackorders type="Int">49</nAllowBackorders>
  <nAllowSubstitution type="Int">48</nAllowSubstitution>
</detail>
<detail date="10302001" name="XAPICall" time="09:56:49" type="" DSTMPL="DXAPIROUTE"
executionOrder="3" parameterCount="4">
  <ClientPort type="Int">6009</ClientPort>
  <ClientIP type="Int">167810863</ClientIP>
  <ClientMagicNumber type="Int">32781408</ClientMagicNumber>
  <XAPIMethodID type="String">GetComputerID</XAPIMethodID>
</detail>
</body>
</event>
</jdeResponse>
```

ルート情報

すべての XAPI イベントについて、上記 XML ファイルの例のグレーの部分に示すように、XML ファイル中に DXAPIROUTE を記述する必要があります。DXAPIROUTE は、送信元クライアントに返されるルート情報を含んでいます。jdeXAPI_Finalize API は、DXAPIROUTE データ実行を追加します。

XAPI イベントの生成

次のリストに、サードパーティに送信する XAPI イベントの生成タスクを示します。

- OCM を設定します。
- インタオペラビリティ・イベント記述テーブル(F90701)内で XAPI イベントを定義します。
- F90702 テーブル内でサブスクリプション情報を設定します。
- サーバーの jde.ini ファイルを構成します。

注:

イベントの生成時に、次のメッセージが表示されることがあります。

> RDEL0000045 – Could not open the tables for reliable event delivery (F90703 and F90704).(高信頼イベント配信のテーブルがオープンできませんでした(F90703 および F90704)。)
> Reliable event delivery will be disabled.(高信頼イベント配信は無効になります。)

これは単なる警告メッセージです。高信頼イベント配信機能を使用していない場合は無視してください。

XAPI 用の OCM の設定

ERP システムへのインターフェイスが ERP クライアントでない場合は、システム呼出しで IEO カーネルを検索できるように OCM を構成する必要があります。OCM を構成するときには、特定の環境を指定してください。また、2 つの重複したマッピングが同時にアクティブ状態にならないようにしてください。

OCM の構成を行うには、〈オブジェクト・マッピングの改訂〉フォームにアクセスして、[オブジェクト・タイプ]フィールドに"XAPI"と入力します。OCM を構成するときに"XAPI"と入力すると、システム・コー

ルが IEO カーネルを検索できるようになります。OCM が正しく構成されていない場合、システムはエラー・メッセージを生成します。XAPI イベントに対する OCM のエラー・メッセージは、リアルタイム・イベントに対する OCM のエラー・メッセージと同じです。

参照

- イベント処理用に〈Object Configuration Manager〉を設定する方法については、『インタオペラビリティ』ガイドの「リアルタイム・イベント用の OCM の設定」

『CNC インプリメンテーション』ガイドの次のトピックを参照してください。

- オブジェクト構成マネージャの処理
- オブジェクトのマッピング

XAPI イベントの定義

XAPI イベントを定義するには、〈インタオペラビリティ・イベントの定義〉プログラム(P90701)を使用します。XAPI イベントを定義すると、[イベント・カテゴリ]フィールドが[コンテナ]に自動的に更新されます。すべての XAPI イベントでデータ構造体オプションが使用されます。XAPI イベントに必要な DXAPIROUTE データ構造体は、自動的に追加されます。DXAPIROUTE データ構造体は、送信元システムに返されるルート情報を含んでいます。jdeXAPI_Finalize API は、DXAPIROUTE データ実行を追加します。XAPI イベントを定義した後は、必ず状況を変更してイベントをアクティブ化してください。

参照

- インタオペラビリティ・イベント記述テーブル(F90701)に XAPI イベントを追加する方法については、『インタオペラビリティ』ガイドの「イベントの定義」

XAPI サブスクリプション情報の設定

XAPI イベントを生成する場合は、論理サブスクライバを定義し、XAPI イベント・サブスクライバ情報を設定する必要があります。論理サブスクライバを定義しなければ、XAPI イベント・サブスクライバ情報を追加できません。サブスクライバ情報がなくても XAPI イベントは生成されますが、配信できません。〈インタオペラビリティ・イベント・サブスクリプション〉プログラム(P90702)を使用して、論理サブスクライバを定義し、XAPI サブスクライバ情報を設定します。XAPI サブスクライバを設定した後は、必ず状況を変更してサブスクライバをアクティブ化してください。

参照

- 論理サブスクライバを定義して XAPI サブスクライバ情報を設定する方法については、『インタオペラビリティ』ガイドの「サブスクライバ情報の設定」

XAPI イベントの jde.ini ファイルの構成

XAPI イベントを生成するには、エンタープライズ・サーバーの jde.ini ファイルの次のセクションを構成する必要があります。

- [JDENET_KERNEL_DEF19]
- [JDENET_KERNEL_DEF20]
- [JDEITDRV]

jde.ini ファイルが XAPI イベント用に正しく構成されていない場合、次のエラー・メッセージが jde.log ファイルに書き込まれます。

> XAPI Event [Event Name] cannot be subscribed.（XPAI イベント[イベント名]はサブスクライブできません。）Must have XAPI Definition in the INI file.（INI ファイルに XAPI 定義を記述する必要があります。）

単一イベントおよびコンテナ・イベント定義テーブル(F90701)で XAPI イベントを定義し、jde.ini ファイルで XAPI エグゼキュータ情報を定義してください。

XAPI サブスクリプションは継続され、解除することができないため、次のエラー・メッセージは無視してください。

> Cannot unsubscribe XAPI event.（XAPI イベントのサブスクリプションを解除できません。）

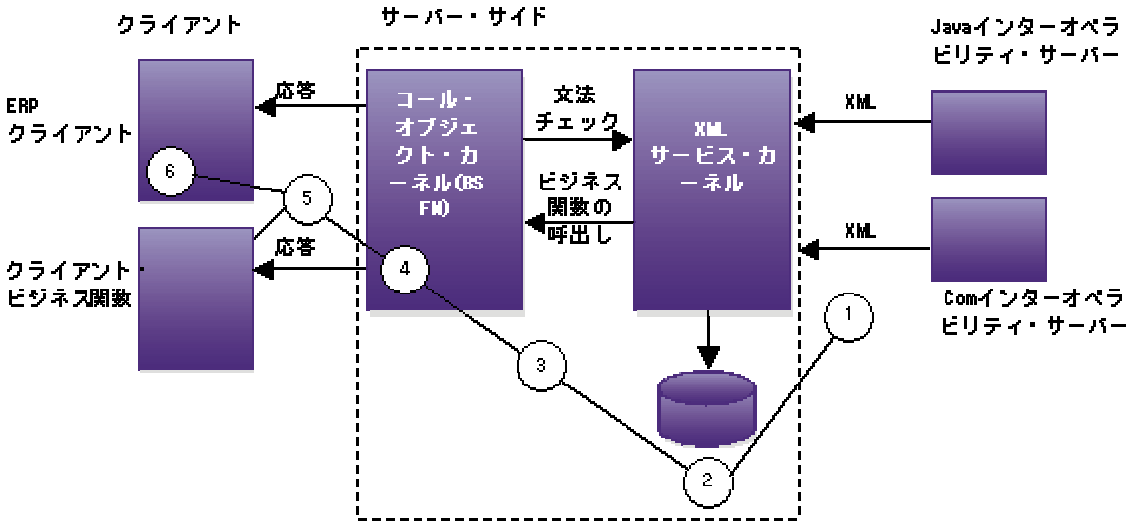
参照

- サーバーのカーネルと[JDEITDRV]の構成については、『インタオペラビリティ』ガイドの「イベントの jde.ini ファイルの構成」

XAPI 受信応答

XAPI 構造は、受信応答用です。XAPI イベントが生成された後で、XAPI 受信応答が発生します。XAPI 受信応答はサードパーティ・システムで処理されます。XAPI の実行者であるサードパーティ・システムは、要求（イベント）を処理して XAPI 入力者に返信を戻します。

戻り XML ドキュメントを受信すると、ドキュメントは XML サービス・カーネルに送られます。XML サービス・カーネルは、XML ドキュメントをディスクに保存し、ユニークなハンドルを作成し、XML ドキュメントの DXAPIROUTE XAPI method ID 要素で指定されたコールバック・ビジネス関数を呼び出します。次の図に、サードパーティ・システムから ERP 送信元システムへの XAPI 応答の流れを示します。点線がフローを表します。



XAPI 関数の受信部分は、次のようになっています。

1. 受信 XML ドキュメントが、サードパーティ・システムから XML サービス・カーネルに渡されます。
2. XML サービス・カーネルがユニークな XML ハンドルを作成し、ドキュメントをディスクに保管します。
3. XML サービス・カーネルが XML ドキュメントから XAPICallMethod 属性を読み取り、指定されたビジネス関数にパラメータとして XML ハンドルを渡します。
4. ビジネス関数(XAPICallMethod)は、XML サービス API を使用して XML データを読み取り、ERP データへと解析します。
5. ビジネス関数(XAPICallMethod)は、XML CallObject を使用して入力者に返信を送ります。
6. ERP クライアントは、ERP サーバーからの XAPI 応答をポーリングできます。

XAPI 応答解析 API

受信 XAPI 応答の生成には次に挙げる API が利用可能です。

- jdeXML_GetDSCount
- jdeXML_GetDSName
- jdeXML_ParseDS
- jdeXML_DeleteXML

注:

XML サービス・カーネルについては、オンライン API ドキュメンテーションを参照してください。

例:XAPI 応答のオペランド解析 API の使用

次の例に、ビジネス関数が XML サービス API を使用して XML データの読取りとオペランド解析を行う方法を示します。

```
int   iCurrentRecord;
      int                  iHeaderCount;

      DSD4205030A                dsD4205030A = {0};
      DSD4205030B                dsD4205030B = {0};

      #ifdef jdeXAPI_CALLS_ENABLED

      if(jdeXAPI_IsCallTypeEnabled("XAPIOPOUT") &&
                  jdeXAPI_IsCallTypeEnabled("XAPIOPIN") )
      {
                  iRecordCount = jdeXML_GetDSCount(lpDS->szXMLHandle);

                  if (iRecordCount > 0)
                  {
                              for (iCurrentRecord = 0; iCurrentRecord < iRecordCount; iCurrentRecord++)
```

```
                    {
                        jdeXML_GetDSName(lpDS->szXMLHandle,
                                                    iCurrentRecord,
                                                    nidDSName);
                    if (jdestrcmp(nidDSName,(const char*)"D4205030A") == 0)
                        {
                            jdeXML_ParseDS( lpDS->szXMLHandle,
                                                    iCurrentRecord,
                                                    &dsD4205030A,
                                                    sizeof(DSD4205030A));
                        }
                    else
                        {
                            jdeXML_ParseDS( lpDS->szXMLHandle,
                                                iCurrentRecord,
                                                &dsD4205030B,
                                                sizeof(DSD4205030B));
                        }

                    }

            }
            if (iCurrentRecord == iRecordCount)
            {
            jdeXML_DeleteXML(lpDS->szXMLHandle);
            }
    }
    #endif
```

例:受信 XAPI 応答

次の例に、受信 XAPI 応答を示します。

```
<?xml version="1.0" encoding="utf-8" ?>
<jdeRequest pwd="JDE" type="xapicallmethod" user="JDE" session="" environment="DV7333"
sessionidle="">
<header>
<eventVesrion>1.0</eventVesrion>
<type>XAPIOPIN</type>
<user>JDE</user>
<application>XPI</application>
<version />
<sessionID />
<environment>DEVXPINT</environment>
<host>denxpi7</host>
<sequenceID />
<date>09122001</date>
<time>094951</time>
<scope />
<codepage>utf-8</codepage>
</header>
<body elementCount="3">
```

```
<params type="D4205030A" executionOrder="1" parameterCount="24">
  <param name="type" />
  <param name="dateStamp" />
  <param name="timeStamp" />
  <param name="szRequestId">1|ZJDE0001</param>
  <param name="szBusinessObjective">Maximize_Service</param>
  <param name="mnResultNumber">0.0</param>
  <param name="mnTotalCost">0.0</param>
  <param name="mnTotalDeliveryCost">0.0</param>
  <param name="mnTotalPrice">0.0</param>
  <param name="mnTotalProfit">0.0</param>
  <param name="mnTotalMargin">0.0</param>
  <param name="mnTotalValue">0.0</param>
  <param name="mnLatestLineDate">0.0</param>
  <param name="mnNumberOfBackorders">0.0</param>
  <param name="mnNumberOfSubstitutions">0.0</param>
  <param name="mnOrderFillRate">0.0</param>
  <param name="szErrorCode" />
  <param name="szErrorDescription" />
  <param name="szOrderNumber">3115|SO|00200</param>
  <param name="nAllowPartialOrderShip">0</param>
  <param name="nAllowMultisource">0</param>
  <param name="nAllowBackorders">0</param>
  <param name="nAllowSubstitution">0</param>
  <param name="nAllowPartialLineShip">0</param>
</params>
<params type="D4205030B" executionOrder="2" parameterCount="28">
  <param name="type" />
  <param name="dateStamp" />
  <param name="timeStamp" />
  <param name="mnLineNumber">1.0</param>
  <param name="mnOriginalLineNumber">1.0</param>
  <param name="mnCacheLineNumber">1.0</param>
  <param name="mnRequestedItem">60011.0</param>
  <param name="mnAvailableItem">60011.0</param>
  <param name="mnAvailableAmount">25.0</param>
  <param name="jdAvaiableDate">09/12/2001 00:00:00</param>
  <param name="jdRequestedDate">09/10/2001 00:00:00</param>
  <param name="jdPickDate">09/11/2001 00:00:00</param>
  <param name="jdShipDate">09/11/2001 00:00:00</param>
  <param name="szShipLocation" />
  <param name="mnCost">0.0</param>
  <param name="mnDeliveryCost">0.0</param>
  <param name="mnPrice">0.0</param>
  <param name="mnProfit">0.0</param>
  <param name="mnMargin">0.0</param>
  <param name="mnValue">0.0</param>
  <param name="mnSubstitutionRatio">0.0</param>
  <param name="szShippingGroup" />
  <param name="szMultiSource" />
  <param name="szErrorCode" />
  <param name="szSuspectedCause" />
  <param name="nAllowPartialOrderShip">0</param>
  <param name="nAllowBackorders">0</param>
  <param name="nAllowSubstitution">0</param>
```

```
</params>
<params type="DXAPIROUTE" executionOrder="3" parameterCount="7">
  <param name="type" />
  <param name="dateStamp">09/05/2001 00:00:00</param>  <param name="timeStamp">13:54:04</param>
  <param name="ClientPort">6009</param>
  <param name="ClientIP">168045665</param>
  <param name="ClientMagicNumber">3</param>
  <param name="XAPIMethodID">OrderPromiseCallback</param>
</params>
</body>
</jdeRequest>
```

XAPI 応答の jde.ini ファイルの構成

エンタープライズ・サーバーの jde.ini ファイルの次のセクションを、XAPI 構造の XAPI 応答部に合わせて構成する必要があります。

- [JDENET_KERNEL_DEF22]
- [JDENET_KERNEL_DEF24]
- [XAPI]
- [XMLLookupInfo]

  [XAPI]
  XMLDirectory=c:¥builds¥bdev¥log¥

  注:
  サーバー上で、jde.ini ファイル内の[XPAI]セクションの XMLDirectory キーに、XML ドキュメントのディレクトリを登録する必要があります。キーには、XML ドキュメントを保存するサーバー上のディレクトリが記述されています。

  [XMLLookupInfo]
  XMLRequestType5=realTimeEvent
  XMLKernelMessageRange5=14251
  XMLKernelHostName5=local
  XMLKernelPort5=0
  XMLKernelReply5=0

参照

- カーネルの構成については、『インタオペラビリティ』ガイドの「イベントの jde.ini ファイルの構成」

XAPI のクライアント jde.ini ファイルの構成

ERP クライアントを使用して XAPI イベントを生成する場合は、クライアント・ディスパッチ・カーネルと、クライアント jde.ini ファイルの[JDENET]セクションを定義する必要があります。エンタープライズ・サーバーへのインターフェイスが ERP クライアントでなければ、この 2 つの設定は不要です。この設定を行うと、ERP クライアントで ERP サーバーからの XAPI 応答メッセージをポーリングできるようになります。

次の設定を使用して、ERP クライアントの jde.ini ファイルを構成します。

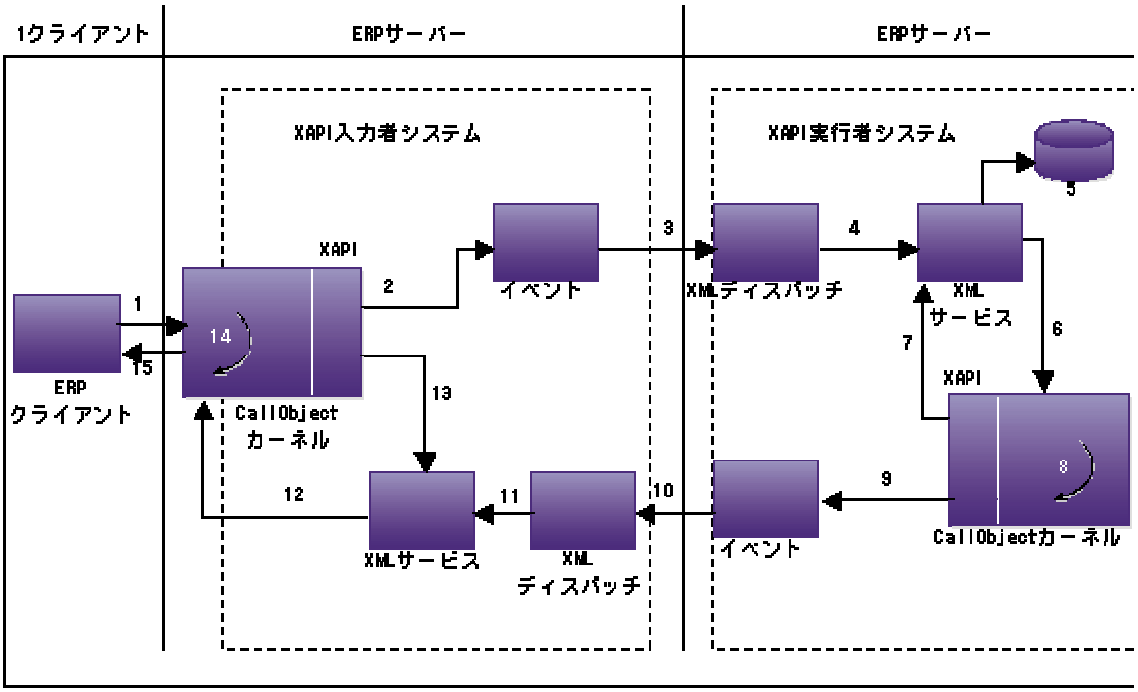
```
[JDENET_KERNEL_DEF27]
krnlName=CLIENT DISPATCH KERNEL
dispatchDLLName=jdeuser.dll
dispatchDLLFunction=_JDENET_ClientDispatch
maxNumberOfProcesses=0
numberOfAutoStartProcesses=0

[JDENET]
serviceNameListen=6004
serviceNameConnect=6004
maxKernelRanges=27
netTrace=0
```

注:
serviceNameListen および serviceNameConnect 設定は、サーバー側と同じ設定にする必要があります。たとえば、サーバーの jde.ini ファイルで serviceNameListen=6005 および serviceNameConnect=6005 に設定している場合は、ERP クライアントの jde.ini ファイルでも serviceNameListen=6005 および serviceNameConnect=6005 に設定する必要があります。

maxKernelRanges 設定の値は、サーバー側と同じにしてください。

ERP システム間の XAPI

XAPI 構造は、2 つの異なる ERP システムが相互に通信するための機能も提供します。ERP システム 1（XAPI 入力者システム）は、XAPI 要求（イベント）を生成します。要求は、サードパーティ・システムに配布される代わりに、JDENet により ERP システム 2 に送られます。XAPI の実行者である ERP システム 2 は、要求を処理して ERP システム 1 に応答を送信します。次の図に、ある ERP システムから他の ERP システムへの XAPI 処理のロジックを示します。



XAPI 入力者のシステム・フロー

1. ERP ビジネス関数が、CallObject カーネル内でインタオペラビリティ・イベント・インターフェイスを呼び出して要求を送信します。
2. ビジネス関数が XAPI API を使用して XAPI 要求を作成します。XAPI がコールバック関数を追加して、要求をイベント構造に送ります。
3. IEO カーネルが XML フォーマットで XAPI イベントを作成し、XML ドキュメントを EVN カーネルに送ります。EVN カーネルが XML ドキュメントを ERP システム 2 の XML ディスパッチ・カーネルに送ります。XML ドキュメントは、持続的なサブスクリプション情報を使用して JDENet を介して送られます。送信者のサーバーおよびポート情報を含むルーティング・トークンが追加されます。イベントのメッセージ・タイプは *RealTimeEvent* であることが必要です。

XAPI 実行者システム

1. XML ディスパッチ・カーネルは XML パッケージを受け取り、イベント要求とルーティング情報を XML サービス・カーネルに送ります。
2. XML サービス・カーネルは XAPI 要求を保管し、それに対するファイル・ハンドルを作成します。また、XML カーネルは XML ベースのルーティング情報も作成して保管し、それに対するファイル・ハンドルを作成します。XML サービス・カーネルはイベント要求定義テーブル(F907012)を使用して、要求を処理するビジネス関数を検索します。
3. XML サービス・カーネルは、XML 要求ハンドルとルーティング情報ハンドルを使用してビジネス関数を(CallObject 内)呼び出します。
4. ビジネス関数が XAPI API を使用して XAPI 要求を解析し、処理します。XAPI API が XML 要求をメモリにロードします。

5. ビジネス関数が XAPI イベント要求を処理します。ビジネス関数が XAPI 応答も作成します。応答のメッセージ・タイプは *xapicallmethod* であることが必要です。XAPI 応答は XML フォーマットです。ビジネス関数は、ルーティング情報ハンドルを解析します。
6. XAPI 応答の入力者が、応答とルーティング情報をイベント構造に送ります。
7. IEO カーネルが XML フォーマットで XAPI 応答を作成し、XML ドキュメントを EVN カーネルに送ります。EVN カーネルがダイレクト・ルーティングを使用して、応答およびルーティング情報を ERP システム 1（XAPI 入力者システム）の XML ディスパッチ・カーネルに送ります。ダイレクト・ルーティングとは、XAPI の返信を同じ要求発信サーバーに送ることを意味します。

XAPI 入力者システム

1. XML ディスパッチ・カーネルが XML ドキュメントを受信し、応答を XML サービス・カーネルに送ります。
2. XML サービス・カーネルは応答ドキュメントを保管し、ファイル・ハンドルを作成し、ファイル・ハンドルを指定してコールバック・ビジネス関数を呼び出します。
3. ビジネス関数は、XAPI API（XAPI 応答ハンドラ）を使用して応答ドキュメントを解析します。XAPI API は XML サービス・カーネルを使用して、ドキュメントをメモリにロードします。
4. ビジネス関数が XAPI API（CallObject カーネル）を使用して応答を処理します。
5. ビジネス関数は、ERP サーバーからの XAPI 応答をポーリングできます。

注:
ある ERP システムから別の ERP システムに要求を送信し、返信なしで処理させることができます。応答を必要としない場合は、要求を処理せずに上記のステップ 8 までを使用します。応答は生成されません。

J.D. Edwards の高信頼配信機能を使用して XAPI イベントを処理できます。

ERP システム間の XAPI API

ここでは、2 つの ERP システムを処理する際に XAPI イベントの生成に使用する API の関数別リストを示します。これらの API については、オンライン API ドキュメンテーションを参照してください。

XAPI 送信要求生成 API

次の API を使用して XAPI イベント（入力者システムからの要求）を生成します。次の各 API は、「XAPI 送信イベント」に示した API と同じです。

- jdeXAPI_SimpleSend
- jdeXAPI_Init
- jdeXAPI_Add
- jdeXAPI_Finalize
- jdeXAPI_Free

参照

『インタオペラビリティ』ガイドの次の例を参照してください。

- 例：XAPI イベント作成 API の使用
- 例：XAPI 送信イベントの XML

XAPI 送信要求処理 API

次の API は、マップされているビジネス関数（実行者システム）で送信 XAPI 要求ドキュメントから XML データを取り込むために使用されます。

- jdeXMLRequest_GetDSCount
- jdeXMLRequest_GetDSName
- jdeXMLRequest_ParseDS
- jdeXMLRequest_DeleteXML
- jdeXMLRequest_ParseNextDSByName
- jdeXMLRequest_PrepareDSListForIterationByName

例：XAPI 送信要求解析 API の使用

次のサンプル・コードに、ERP 入力者システムから ERP 実行者システムへの送信要求を生成する API の使用例を示します。

```
API_System FunctionsSampleXAPIRequestParsingAPIUsage
Last Modified: ERP 9.0| October 21, 2002

Example
int          iXMLRecordCount  = 0;
int          iCurrentRecord   = 0;
NID           nidDSName;
ID           idReturnValue    = ER_SUCCESS;
ID           idSORecordCount  = ER_ERROR;  /*Return Code*/
MATH_NUMERIC  mnBatchNumber    = {0};
unsigned long lBatchNumber     = {0};

DSD4206030A   dsD4206030A      = {0};
/* CacheProcessInboundDemandRequest B4206030.c */
DSD4206000I   dsD4206000I      = {0};
/* Demand scheduling inbound DSTR */


iXMLRecordCount  =  jdeXMLRequest_GetDSCount(lpDS->szXMLHandle);

if( iXMLRecordCount > 0)
{
  for ( iCurrentRecord = 0; iCurrentRecord < iXMLRecordCount; iCurrentRecord++)
  {
     memset((void *)(&dsD4206000I), (int)(_J('¥0')), sizeof(DSD4206000I));
```

```
    memset((void *)(nidDSName), (int)(_J('¥0')), sizeof(NID));

    if(jdeXMLRequest_GetDSName(lpDS->szXMLHandle,
                               iCurrentRecord,
                               nidDSName))
    {
      /* Retrieving data*/
      if (jdeStricmp(nidDSName, (const JCHAR *)_J("D40R0180B")) == 0)
{
        if (jdeXMLRequest_ParseDS(lpDS->szXMLHandle,
                               iCurrentRecord,
                               &dsD4206000I,
                               sizeof(DSD4206000I)))
        {
          /* Get next number for the batch number of the
         inbound INVRPT record*/if
         ( dsD4206000I.cInventoryAdvisement == _J('1'))
         {
           lBatchNumber    = JDB_GetInternalNextNumber();
           LongToMathNumeric(lBatchNumber, &mnBatchNumber);
           FormatMathNumeric(dsD4206000I.szBatch,
&mnBatchNumber);
         }

         /* Setup cancel flag for pending delete record */
         if ( dsD4206000I.cPendingDelete == _J('1'))
         { /* Flag set as 1 for any cancel demand record */
         dsD4206000I.cCancelFlag = _J('1');
         }
         else
         { /* Flag set as 9 for any non cancel demand record */
         dsD4206000I.cCancelFlag = _J('9');
         }

         /* Load parms for cache */
         memset((void *)(&dsD4206030A), (int)(_J('¥0')), sizeof(DSD4206030A));

        I4206000_LoadParmsToCache(&dsD4206000I, &dsD4206030A);
        MathCopy(&dsD4206030A.mnJobnumberA, lpmnJobNumber);

        /* Add the DSTR to cache */
        idReturnValue = jdeCallObject( _J("CacheProcessInboundDemandRequest") ,
                               (LPFNBHVR)NULL ,
                               lpBhvrCom ,
                               lpVoid ,
                               (LPVOID)&dsD4206030A,
                               (CALLMAP *) NULL,
                               (int) 0,
                               (JCHAR*)NULL ,
                               (JCHAR*)NULL ,
                               (int) 0 );
        /* Write XML DSTR to cache fail */
        if (idReturnValue == ER_ERROR)
        {
          jdeErrorSet(lpBhvrCom, lpVoid, (ID) 0, _J("032E"), (LPVOID) NULL);
```

```
            }

          }
          else
          {   /* warning XML parse fail */
                jdeErrorSet(lpBhvrCom, lpVoid, (ID) 0, _J("40R46"), (LPVOID) NULL);
          }
        } /* end if  */
     }/* end if DS name */
   }/* end for - looping all matching XML DSTR */

   /* Ensure there is at least one record */
   idSORecordCount = ER_SUCCESS;

}/*if( iXMLRecordCount > 0) */

return idSORecordCount;
```

例:ERP 入力者システムからの XAPI 要求の XML

次の例に、ERP 入力者システムから ERP 実行者システムへの XAPI 要求ドキュメントを示します。

```
<?xml version="1.0" encoding="UTF-16" ?>
<jdeRequest pwd="4f3e65076f446c5d20666f4172536518435c" role="*ALL" type="xapicallmethod"
user="PP6954083" session="" environment="DV9NIS2" responseCreator="XAPI">
  <header>
    <eventVersion>1.0</eventVersion>
    <type>XAPIDEMO</type>
    <user>PP6954083</user>
    <role>*ALL</role>
    <application />
    <version />
    <sessionID>35087181.1050101193</sessionID>
    <environment>DV9NIS2</environment>
    <host>DEN-PP6954083B</host>
    <sequenceID>DEN-PP6954083B_3112_041120031647161</sequenceID>
    <date>04112003</date>
    <time>164716</time>
    <scope />
    <codepage>utf-8</codepage>
    <instanceInfo>
      <host>DEN-PP6954083B</host>
      <port>6025</port>
      <type>JDENET</type>
    </instanceInfo>
  </header>
  <body elementCount="3">
    <errors errorCount="4">
      <error code="041H" type="BSFN ERROR" />
      <error code="041I" type="BSFN ERROR" />
      <error code="2597" type="BSFN ERROR" />
      <error code="4136" type="BSFN ERROR" />
```

```
   </errors>
   <params type="D907001A" executionOrder="0" parameterCount="14">
    <param name="szXMLHandle">DEN-PP6954083B_||_C:¥builds¥B9_SP0¥log¥J3E9745EE032D-00000C28-
00000001-00000000000000000000FFFF0A0396A3.xml</param>
    <param name="mnAddressNumber">55617</param>
    <param name="szNameAlpha">Pradip Pandey</param>
    <param name="szNameMailing">Pradip K Pandey</param>
    <param name="szAddressLine1" />
    <param name="szAddressLine2" />
    <param name="szZipCodePostal">80237</param>
    <param name="szCity">Denver</param>
    <param name="szState">CO</param>
    <param name="szCountry" />
    <param name="mnAmountGross">100.00</param>
    <param name="mnUnits">100.00</param>
    <param name="jdDtForGLAndVouch1">2001/01/01</param>
    <param name="cDefaultAddressLine1">9</param>
   </params>
   <params type="DXAPIROUTE" executionOrder="1" parameterCount="4">
    <param name="ClientPort">6024</param>
    <param name="ClientIP">168007331</param>
    <param name="ClientMagicNumber">1</param>
    <param name="XAPIMethodID">XAPITestResponse</param>
   </params>
  </body>
</jdeRequest>
```

XAPI 受信応答生成 API

次の API は、応答の生成に使用されます（実行者システム）。

- jdeXAPIResponse_SimpleSend
- jdeXAPIResponse_Init
- jdeXAPIResponse_Add
- jdeXAPIResponse_Finalize
- jdeXAPIResponse_Free

例：XAPI 受信応答解析 API の使用

次のサンプル・コードに、ERP 実行者システムから ERP 入力者システムへの受信返信を生成する API の使用例を示します。

```
JDEBFRTN (ID) JDEBFWINAPI SendOrderPromiseRequest (LPBHVRCOM lpBhvrCom, LPVOID lpVoid,
LPDSD4205010 lpDS)
{
  /************************************************************************
* Variable declarations
************************************************************************/

    char       cPromisableLine            = ' ';
```

```
    int        nHeaderBackOrderAllowed     = ' ';
    HUSER      hUser       ;
    ID         JDEDBResult                 = JDEDB_PASSED;
    BOOL       bExit                       = FALSE;
    BOOL       bB4001040Called             = FALSE;
    BOOL       bXAPIInUse                  = FALSE;
    BOOL       bAtLeastOneDetail           = FALSE;

    #ifdef jdeXAPI_CALLS_ENABLED
    XAPI_CALL_ID          ulXAPICallID      = 0;
    XAPI_CALL_RETURN      eXAPICallReturn   = eEventCallSuccess;
    #endif
/*****************************************************************
* Declare structures
*****************************************************************/
    DSD4001040      dsD4001040        = {0};
    DSD4205020      dsD4205020        = {0};
    DSD4205040      dsD4205040        = {0}; /* Header Info */
    DSD4205050      dsD4205050        = {0}; /* Detail Info */
    DSD4205010A     dsD4205010A       = {0}; /* Query Header */
    DSD4205010B     dsD4205010B       = {0} /* Query Detail */
    DSD0100042      dsD0100042        = {0};
    LPDSD4205040H   lpDSD4205040H         = (LPDSD4205040H) NULL;
    LPDSD4205050D   lpDSD4205050D         = (LPDSD4205050D) NULL;


   /*****************************************************************
* Declare pointers
*****************************************************************/

/*****************************************************************
* Check for NULL pointers
*****************************************************************/
if ((lpBhvrCom == (LPBHVRCOM) NULL) ||
   (lpVoid    == (LPVOID)    NULL) ||
   (lpDS      == (LPDSD4205010)             NULL))
{
  jdeErrorSet (lpBhvrCom, lpVoid, (ID) 0, "4363", (LPVOID) NULL);
  return ER_ERROR;
}

/* Retrieving hUser */

JDEDBResult = JDB_InitBhvr (lpBhvrCom, &hUser, (char *)NULL, JDEDB_COMMIT_AUTO ) ;

if ( JDEDBResult == JDEDB_FAILED )
{
  jdeSetGBRError ( lpBhvrCom, lpVoid, (ID) 0, "4363" ) ;
  return ER_ERROR ;
}
/*****************************************************************
* Set pointers
*****************************************************************/
/*****************************************************************
* Main Processing
```

```
**********************************************************************/
   /*-------------------------------------------------------*/
   /* Setting Up ErrorCode                                  */
            */
   lpDS->cErrorCode = '0';

   /*-------------------------------------------------------*/
   /* Determing if XAPI is ready to be used                 */

   bXAPIInUse = FALSE;

   #ifdef jdeXAPI_CALLS_ENABLED
   if(jdeXAPI_IsCallTypeEnabled("XAPIOPOUT") &&
          jdeXAPI_IsCallTypeEnabled("XAPIOPIN") )
   {
          bXAPIInUse = TRUE;
   }
   #endif
   /*-------------------------------------------------------*/
   /* Data validation and defaults.                         */
   /* When Display Before Accecpt Mode is on,validate Key */
   /* Information.Otherwise retrieve it from Header Record*/

   if((lpDS->cDisplayBeforeAcceptMode == '1')      &&
            ((MathZeroTest(&lpDS->mnOrderNumber) == 0) ||
              (IsStringBlank(lpDS->szOrderType))      ||
              (IsStringBlank(lpDS->szOrderCompany))))
   {
           bExit = TRUE;
   }
   else
   {
          MathCopy(&dsD4205040.mnOrderNumber,&lpDS->mnOrderNumber);
          strncpy(dsD4205040.szOrderType,
            lpDS->szOrderType,
               sizeof(dsD4205040.szOrderType));
          strncpy(dsD4205040.szComputerID,
            lpDS->szOrderCompany,
               sizeof(dsD4205040.szOrderCompany));
          dsD4205040.cUseCacheOrWF = lpDS->cUseCacheOrWF;
          strncpy(dsD4205040.szComputerID,
            lpDS->szComputerID,
               sizeof(dsD4205040.szComputerID));
          MathCopy(&dsD4205040.mnJobNumber,&lpDS->mnJobNumber);
          jdeCallObject( "GetSalesOrderHeaderRecord",
                    NULL,
                    lpBhvrCom, lpVoid,
                    (LPVOID)&dsD4205040,
                    (CALLMAP *) NULL,
                    (int) 0,
                    (char *) NULL,
                    (char *) NULL,
                    (int) 0 ) ;

          lpDSD4205040H = (LPDSD4205040H)jdeRemoveDataPtr(hUser,
```

```
(ulong)dsD4205040.idHeaderRecord);

        if (lpDSD4205040H == NULL)
        {
                bExit = TRUE;
        }
  }

  /*-------------------------------------------------------*/
  /* Set error if we're exiting at this point            */

  if (bExit == TRUE)
  {
        lpDS->cErrorCode = '1';
        /* Sales Order Header Not Found */
        strncpy(lpDS->szErrorMessageID,
             "072T",
                sizeof(lpDS->szErrorMessageID));
        if (lpDS->cSuppressError != '1')
        {
             jdeErrorSet (lpBhvrCom, lpVoid, (ID) 0, "072T", (LPVOID) NULL);
        }
  }

  /*-------------------------------------------------------*/
  /* Default Promising Flag is always 1 for Xe           */
  lpDS->cDefaultPromisingFlags = 1;

  if (bExit == FALSE)
  {
    /*-------------------------------------------------------*/
    /* Call XAPIInit      */

    #ifdef jdeXAPI_CALLS_ENABLED
    if(bXAPIInUse == TRUE)
    {
        ulXAPICallID = jdeXAPI_Init( lpBhvrCom,
                              "SendOrderPromiseRequest",
                        "XAPIOPOUT",
                        NULL,
                        &eXAPICallReturn);
        if (eXAPICallReturn != eEventCallSuccess)
        {
                bExit = TRUE;
        }
    }
    #endif
    if (bExit == FALSE)
    {
        /*-------------------------------------------------*/
        /* Loading Header Information                    */

        I4205010_PopulateQueryHeader(lpDS,&dsD4205010A,
          lpDSD4205040H,&dsD0100042,hUser,lpVoid,lpBhvrCom);
```

```
        nHeaderBackOrderAllowed = dsD4205010A.nAllowBackorders;

        /*-------------------------------------------------*/
        /* Adding Header Information          */

        #ifdef jdeXAPI_CALLS_ENABLED
        if(bXAPIInUse == TRUE)
        {
            eXAPICallReturn = jdeXAPI_Add( lpBhvrCom,
                                ulXAPICallID,
                                "SendOrderPromiseRequest",
                                "D4205010A",
                                &dsD4205010A,
                                sizeof(DSD4205010A));
            if (eXAPICallReturn != eEventCallSuccess)
            {
                bExit = TRUE;
            }
        }
        #endif
    }
}

if (bExit == FALSE)
{
    /*-------------------------------------------------*/
    /* Loading Detail Information              */

    MathCopy(&dsD4205050.mnOrderNumber,&lpDS->mnOrderNumber);
    strncpy(dsD4205050.szOrderType,lpDS->szOrderType,
                        sizeof(dsD4205050.szOrderType));
    strncpy(dsD4205050.szOrderCompany,lpDS->szOrderCompany,
                        sizeof(dsD4205050.szOrderCompany));
    dsD4205050.cUseCacheOrWF = lpDS->cUseCacheOrWF;
    strncpy(dsD4205050.szComputerID,lpDS->szComputerID,
                        sizeof(dsD4205050.szComputerID));
    MathCopy(&dsD4205050.mnJobNumber,&lpDS->mnJobNumber);
    if (lpDSD4205040H->cActionCode != 'A')
    {
        dsD4205050.cCheckTableAfterCache = '1';
    }
    else
    {
        dsD4205050.cCheckTableAfterCache = '0';
    }
    jdeCallObject( "GetSalesOrderDetailRecordOP",
        NULL,
        lpBhvrCom, lpVoid,
        (LPVOID)&dsD4205050,
        (CALLMAP *) NULL,
        (int) 0, (char *) NULL,
        (char *) NULL, (int) 0 ) ;

    if (dsD4205050.cRecordFound != '1')
    {
```

```
        bExit = TRUE;
        lpDS->cErrorCode = '1';
        /* Sales Order Detail Not Found */
        strncpy(lpDS->szErrorMessageID,"4162",
                        sizeof(lpDS->szErrorMessageID));
        if (lpDS->cSuppressError != '1')
        {
             jdeErrorSet (lpBhvrCom, lpVoid, (ID) 0, "4162", (LPVOID) NULL);
        }
    }

    while ((dsD4205050.cRecordFound == '1') && (bExit == FALSE))
    {
        lpDSD4205050D = (LPDSD4205050D)jdeRemoveDataPtr( hUser, (ulong)dsD4205050.idDetailRecord);
        /* Reset flags */
        cPromisableLine = '0';
        bB4001040Called = FALSE;

        /*-----------------------------------------------*/
        /* Evaluate the Record from F4211 (cDataSource = 2)*/
        /*   to find out if we should promise the line */
        /*   else find out from Order Promising Detail.*/

        if(dsD4205050.cDataSource == '1')
        {
            if (lpDSD4205050D->cOPPromiseLineYN == 'Y')
            {
                cPromisableLine = '1';
            }
        }
        else if(dsD4205050.cDataSource == '2')
        {
            MathCopy ( &dsD4001040.mnShortItemNumber,
                       &lpDSD4205050D->mnShortItemNumber);
            strncpy ( dsD4001040.szBranchPlant,
                 lpDSD4205050D->szBusinessUnit,
                      sizeof(dsD4001040.szBranchPlant));

            jdeCallObject   ( "GetItemMasterDescUOM",
                              NULL,
                              lpBhvrCom, lpVoid,
                               (LPVOID)&dsD4001040,
                               (CALLMAP *) NULL,
                               (int) 0, (char *) NULL,
                               (char *) NULL, (int) 0 ) ;

            bB4001040Called = TRUE;

            cPromisableLine = I4205010_IsLinePromisable(lpBhvrCom,lpVoid,
                                hUser,lpDS,lpDSD4205050D, dsD4001040.cStockingType);
        }
        if (cPromisableLine == '1')
        {
          /* Set this flag if at least one promisable */
          /*      detail record exists. */
```

```
  bAtLeastOneDetail = TRUE;

  if (bB4001040Called == FALSE)
  {
     MathCopy (&dsD4001040.mnShortItemNumber,
               &lpDSD4205050D->mnShortItemNumber);
     strncpy ( dsD4001040.szBranchPlant,
               lpDSD4205050D->szBusinessUnit,
               sizeof(dsD4001040.szBranchPlant));

  jdeCallObject   ( "GetItemMasterDescUOM",
                    NULL,
                    lpBhvrCom, lpVoid,
                    (LPVOID)&dsD4001040,
                    (CALLMAP *) NULL,
                    (int) 0, (char *) NULL,
                    (char *) NULL, (int) 0 ) ;
  }

  I4205010_PopulateQueryDetail( lpDS,&dsD4205010B,
                        lpDSD4205050D,
                    &dsD4001040,
                    &dsD4205010A,
                        &dsD0100042,
                    cPromisableLine,
                         hUser,
                         lpVoid,
                    lpBhvrCom);

  #ifdef jdeXAPI_CALLS_ENABLED
  if(bXAPIInUse == TRUE)
  {
      eXAPICallReturn = jdeXAPI_Add( lpBhvrCom,
                        ulXAPICallID,
                        "SendOrderPromiseRequest",
                        "D4205010B",
                        &dsD4205010B,
                        sizeof(DSD4205010B));
      if (eXAPICallReturn != eEventCallSuccess)
      {
              bExit = TRUE;
      }
  }
  #endif
}

/*-----------------------------------------------*/
/* Fetching the next Detail Record               */

MathCopy(&dsD4205050.mnOrderNumber,&lpDS->mnOrderNumber);
strncpy(dsD4205050.szOrderType,lpDS->szOrderType,
                 sizeof(dsD4205050.szOrderType));
strncpy(dsD4205050.szOrderCompany,lpDS->szOrderCompany,
                 sizeof(dsD4205050.szOrderCompany));
dsD4205050.cUseCacheOrWF = lpDS->cUseCacheOrWF;
```

```
strncpy(dsD4205050.szComputerID,lpDS->szComputerID,
                sizeof(dsD4205050.szComputerID));
MathCopy(&dsD4205050.mnJobNumber,&lpDS->mnJobNumber);
if (lpDSD4205040H->cActionCode != 'A')
{
     dsD4205050.cCheckTableAfterCache = '1';
     }
     else
     {
          dsD4205050.cCheckTableAfterCache = '0';
     }
     jdeCallObject( "GetSalesOrderDetailRecordOP",
                    NULL,
                    lpBhvrCom, lpVoid,
                    (LPVOID)&dsD4205050,
                    (CALLMAP *) NULL,
                    (int) 0, (char *) NULL,
                    (char *) NULL, (int) 0 ) ;
}

if (!bAtLeastOneDetail)
{
     bExit = TRUE;
     lpDS->cErrorCode = '1';
     /* Sales Order Detail Not Found */
     strncpy(lpDS->szErrorMessageID,"4162",
                sizeof(lpDS->szErrorMessageID));
     if (lpDS->cSuppressError != '1')
     {
          jdeErrorSet (lpBhvrCom, lpVoid, (ID) 0, "4162", (LPVOID) NULL);
     }
}
if (bExit == FALSE)
{
     #ifdef jdeXAPI_CALLS_ENABLED
     if(bXAPIInUse == TRUE)
     {
          eXAPICallReturn = jdeXAPI_Finalize( lpBhvrCom,
                    ulXAPICallID,
                    "SendOrderPromiseRequest",
                    "OrderPromiseCallback");
          if (eXAPICallReturn != eEventCallSuccess)
          {
               bExit = TRUE;
          }
     }
     #endif
}

/*-----------------------------------------------*/
/* Call B4205020 in Add Mode     */

if((bExit == FALSE) &&
      (lpDS->cDisplayBeforeAcceptMode != '1') &&
      (lpDS->cUseCacheOrWF == '2'))
```

```
        {
            MathCopy(&dsD4205020.mnOrderNumber,&lpDS->mnOrderNumber);
            strncpy(dsD4205020.szOrderType,lpDS->szOrderType,
                    sizeof(dsD4205020.szOrderType));
            strncpy(dsD4205020.szOrderCompany,lpDS->szOrderCompany,
                    sizeof(dsD4205020.szOrderCompany));
           strncpy(dsD4205020.szComputerID,lpDS->szComputerID,
                    sizeof(dsD4205020.szComputerID));
          MathCopy(&dsD4205020.mnJobNumber,&lpDS->mnJobNumber);

          jdeCallObject( "MaintainOPWorkFile",
                NULL,
                lpBhvrCom, lpVoid,
                (LPVOID)&dsD4205020,
                (CALLMAP *) NULL,
                (int) 0, (char *) NULL,
                (char *) NULL, (int) 0 ) ;
        }
    }

 /************************************************************************
* Function Clean Up
************************************************************************/
    #ifdef jdeXAPI_CALLS_ENABLED
    if (eXAPICallReturn != eEventCallSuccess)
    {
      /*-------------------------------------------------------*/
      /* CleanUp */

      if(bXAPIInUse == TRUE)
      {
        jdeXAPI_Free( lpBhvrCom,
              ulXAPICallID,
              "SendOrderPromiseRequest");
      }

      lpDS->cErrorCode = '1';
      /* System Error - no reasonable error messages exist */
      /*     in Xe. */
      strncpy(lpDS->szErrorMessageID,"018Y",
                        sizeof(lpDS->szErrorMessageID));
      if (lpDS->cSuppressError != '1')
      {
            jdeErrorSet (lpBhvrCom, lpVoid, (ID) 0, "018Y", (LPVOID) NULL);
      }
    }
    #endif


    if(lpDSD4205040H != (LPDSD4205040H)NULL)
    {
        jdeFree((void *)lpDSD4205040H);
    }

    if(lpDSD4205050D != (LPDSD4205050D)NULL)
```

```
    {
        jdeFree((void *)lpDSD4205050D);
    }
  return (ER_SUCCESS);
}
```

例:入力者システムからの XAPI 応答

次の例に、ERP 実行者システムから ERP 入力者システムへの XAPI 応答ドキュメントを示します。

```
<?xml version="1.0" encoding="UTF-16" ?>
<jdeResponse pwd="4f3e65076f446c5d20666f4172536518435c" role="*ALL" type="realTimeEvent"
user="PP6954083" session="35087181.1050101193" environment="DV9NIS2" responseCreator="XAPI">
 <event>
  <header>
   <eventVersion>1.0</eventVersion>
   <type>XAPIDEMO</type>
   <user>PP6954083</user>
   <role>*ALL</role>
   <application>P90701XT</application>
   <version />
   <sessionID>35087181.1050101193</sessionID>
   <environment>DV9NIS2</environment>
   <host>DEN-PP6954083B</host>
   <sequenceID>DEN-PP6954083B_2864_041120031636402</sequenceID>
   <date>04112003</date>
   <time>164646</time>
   <scope />
   <codepage>utf-8</codepage>
   <instanceInfo>
    <host>DEN-PP6954083B</host>
    <port>6025</port>
    <type>JDENET</type>
   </instanceInfo>
  </header>
  <body elementCount="2">
   <detail date="04112003" name="XAPITestFunctionInitiateRequest" time="16:39:54" type=""
DSTMPL="D907001A" executionOrder="0" parameterCount="14">
   <szXMLHandle type="String" />
   <szNameAlpha type="String">Pradip Pandey</szNameAlpha>
   <szNameMailing type="String">Pradip K Pandey</szNameMailing>
   <szAddressLine1 type="String" />
   <szAddressLine2 type="String" />
   <szZipCodePostal type="String">80237</szZipCodePostal>
   <szCity type="String">Denver</szCity>
   <szState type="String">CO</szState>
   <szCountry type="String" />
   <mnAmountGross type="Double">100.00</mnAmountGross>
   <mnUnits type="Double">100.00</mnUnits>
   <jdDtForGLAndVouch1 type="Date">2001/01/01</jdDtForGLAndVouch1>
   <cDefaultAddressLine1 type="Character" />
  </detail>
```

```
   <detail date="04112003" name="XAPITestFunctionInitiateRequest" time="16:39:54" type=""
DSTMPL="DXAPIROUTE" executionOrder="1" parameterCount="4">
     <ClientPort type="Int">6024</ClientPort>
     <ClientIP type="Int">168007331</ClientIP>
     <ClientMagicNumber type="Int">1</ClientMagicNumber>
     <XAPIMethodID type="String">XAPITestResponse</XAPIMethodID>
   </detail>
  </body>
 </event>
</jdeResponse>
```

XAPI 受信応答処理 API

次の API は、受信 XAPI ドキュメントから XML データを取り込んで、受信 XAPI 応答を生成するために使用されます（入力者システム）。次の各 API は、「XAPI 受信応答」に示した API と同じです。

- jdeXML_GetDSCount
- jdeXML_GetDSName
- jdeXML_ParseDS
- jdeXML_DeleteXML
- jdeXML_ParseNextDSByName
- jdeXML_PrepareDSListForIterationByName

参照

『インタオペラビリティ』ガイドの次の例を参照してください。

- 例：XAPI 応答のオペランド解析 API の使用
- 例：受信 XAPI 応答

XAPI エラー処理 API

次の API は、実行者システムでエラー処理に使用されます。

- jdeXML_CheckSystemError

  システム・エラー・チェック API は、システム・エラー用です。この API は、ERP 実行者システムでシステム・エラーが発生したことを ERP 入力者システムに通知します。

- jdeXML_GetErrorCount
- jdeXML_SetErrors

  エラー・カウント取得 API とエラー設定 API は、ビジネス・エラー用です。この 2 つの API が併用されると、ビジネス・エラーの数を検出して、エラーを解決のために BHVRCOM 構造に送ります。

XAPI の ERP 要求と ERP 応答の生成

ERP 入力者システムからの XAPI 要求と ERP 実行者システムからの返信を生成するには、次のタスクを実行します。

- OCM を設定します。
- XAPI イベントを定義します(F90701)。
- サブスクリプション情報を設定します(F90702)。
- サーバーの jde.ini ファイルを構成します。

注:

OCM の設定、XAPI イベントの定義、およびサブスクリプション情報の設定の各タスクは、『インタオペラビリティ』ガイドの「XAPI 送信イベント」で説明したものと同じです。

ここでは jde.ini ファイルの構成について説明します。

上記の設定タスクに加えて、次のタスクを実行する必要があります。

- ビジネス関数をマップします(F907012)。
- XML ドキュメントの要素名を修正します。

XAPI の ERP 入力者と ERP 実行者の jde.ini ファイルの構成

XAPI イベントを生成するには、エンタープライズ・サーバーの jde.ini ファイルの次のセクションを構成する必要があります。

- [JDENET_KERNEL_DEF19]
- [JDENET_KERNEL_DEF20]
- [JDENET_KERNEL_DEF22]
- [JDENET_KERNEL_DEF24]
- [JDEITDRV]
- [XAPI] - XMLDirectory 設定
- [XMLLookupInfo]
- [INTEROPERABILITY] - LEVEL 設定

```
[XAPI]
XMLDirectory=c:¥builds¥bdev¥log¥

[XMLLookupInfo]
XMLRequestType5=XAPICallMethod
XMLKernelMessageRange5=14251
XMLKernelHostName5=local
XMLKernelPort5=0
XMLKernelReply5=0
```

```
XMLRequestType6=realTimeEvent
XMLKernelMessageRange6=14251
XMLKernelHostName6=local
XMLKernelPort6=0
XMLKernelReply6=0

[INTEROPERABILITY]
LEVEL=DOC
```

注:
LEVEL 設定では、エンタープライズ・サーバーでデバッグ処理のためにイベント XML ドキュメントを記録するレベルを指定します。

LEVEL=DOC キーを設定すると、すべてのリアルタイム・イベントがディスクに書き込まれ、エンタープライズ・サーバーのパフォーマンスが大幅に低下する可能性があります。本稼働用環境や、QA 環境でのストレス・テストには、LEVEL=DOC 設定を使用しないことをお勧めします。

ERP クライアントを使用して XAPI イベントを生成する場合は、クライアント・ディスパッチ・カーネルと、クライアント jde.ini ファイルの[JDENET]セクションを定義する必要があります。

参照

『インタオペラビリティ』ガイドの次のトピックを参照してください。

- エンタープライズ・サーバー用のカーネル構成の設定については、「イベントの jde.ini ファイルの構成」
- ERP クライアント用のカーネル構成と JDENET 構成の設定については、「XAPI のクライアント jde.ini ファイルの構成」

XAPI 要求および応答用のビジネス関数のマッピング

ERP 実行者システムは、ERP 入力者システムからイベントを受け取った時点で、XAPI 要求を処理するために呼び出すビジネス関数またはシステム API を知る必要があります。そこで、ビジネス関数またはシステム API を XAPI イベント名にマップします。この操作は、イベント要求定義テーブル(F907012)内で行います。

ビジネス関数をマップする場合は、ビジネス関数名を入力します。API をマップする場合は、API 名とその定義が含まれているライブラリの名前を入力する必要があります。また、API のシグネチャは、ビジネス関数に類似する共通のものにしてください。

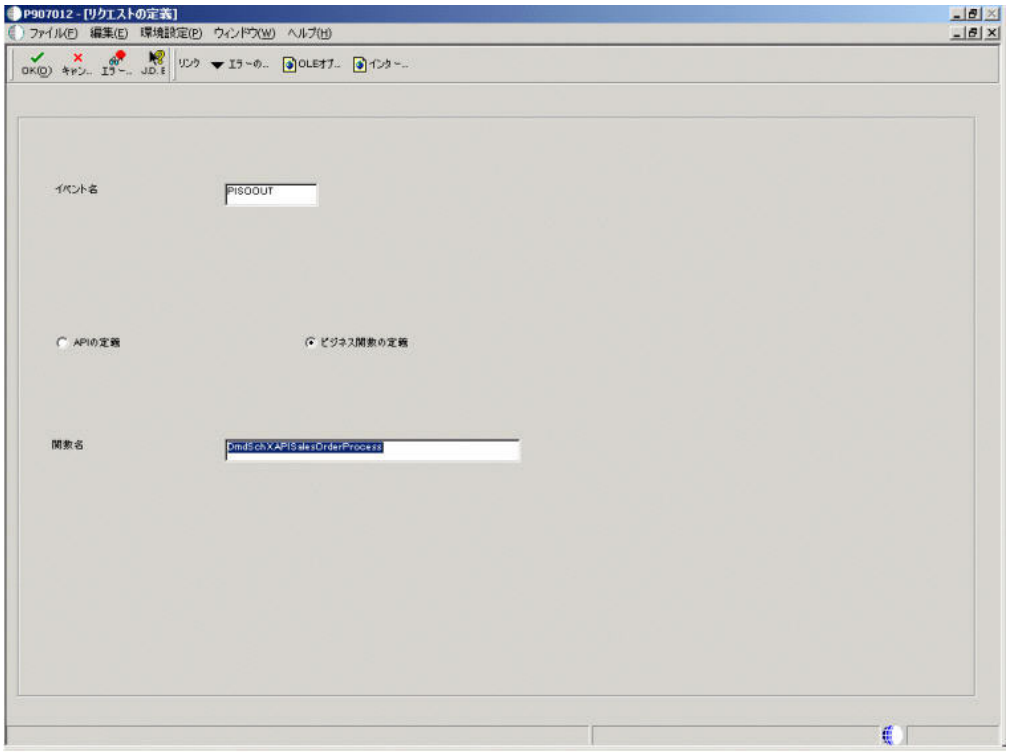
ビジネス関数をマップすると、XAPI イベントでビジネス関数または記述したシステム API を指すことができます。J.D. Edwards 提供のビジネス関数の場合、ソース・コードを修正する必要はありません。

ビジネス関数と API のマッピングには、〈イベント要求定義〉プログラム(P907012)を使用します。

► ハンドラ情報を設定するには

略式コマンド行で P907012 と入力して、〈定義の処理〉フォームにアクセスします。

1. 〈定義の処理〉で[追加]をクリックします。



2. 〈リクエストの定義〉で、次のフィールドに値を入力します。
   - イベント名
   - 関数名
3. 次のオプションのうち 1 つを選びます。
   - API の定義
   - ビジネス関数の定義

注:
[API の定義]を選ぶと、[DLL 名]フィールドが表示されます。[DLL 名]フィールドに値を入力してください。

4. [OK]をクリックして入力を保存します。
5. [キャンセル]をクリックして〈定義の処理〉に戻ります。
6. [閉じる]をクリックしてメイン・メニューに戻ります。

フィールド記述

記述	用語解説
イベント名	イベント名。たとえば、JDERTSOOUT などがあります。単一イベントは別のイベントの一部です。
ビジネス関数の定義	イベントの処理タイプを示すオプション。
API の定義	イベントの処理タイプを示すオプション。
関数名	関数の実際の名前。標準的な ANSI C の命名規則に従います。単語間にスペースは用いません。
DLL 名	データベースドライバ・ファイル名を指定します。このファイルは、エンタープライズ・サーバーの JDE.INI ファイルの[DB SystemSettings (DB システム設定)]で指定します。指定するファイルは、プラットフォームおよびデータベースによって異なります。各マシンおよびデータベースに対する値は次のとおりです。 AS/400 から DB2/400 = DBDR AS/400 から他の DBMS = JDBNET HP9000 から DB2/400 = LIBJDBNET.SL HP9000 から Microsoft SQL Server = LIBJDBNET.SL HP9000 から Oracle (Version 8.0) UNIX = LIBORA80.SL RS6000 から DB2/400 = LIBJDBNET.SO RS6000 から Microsoft SQL Server = LIBJDBNET.SO RS6000 から Oracle (VERSION 7.3) UNIX = LIBORA73.SO RS6000 TO Oracle (VERSION 8.0) UNIX = LIBORA80.SO Intel から AS/400 = JDBODBC.DLL Intel から Oracle (VERSION 7.2) NT = JDBOCI32.DLL Intel から Oracle (VERSION 7.3) NT = JDBOCI73.DLL Intel から Oracle (VERSION 8.0) NT = JDBOCI80.DLL Intel から SQL Server NT = JDBODBC.DLL DEC Alpha から AS/400 = JDBNET.DLL DEC Alpha から Oracle (VERSION 7.2) NT = JDBOCI32.DLL DEC Alpha から Oracle (VERSION 7.3) NT = JDBOCI73.DLL DEC Alpha から Oracle (VERSION 8.0) NT =JDBOCI80.DLL DEC Alpha から SQL Server NT = JDBODBC.DLL

XML ドキュメントの要素名の修正

XAPI ERP 処理までは、ERP システムから発信されたドキュメントは応答ドキュメントと見なされ、ERP システムに着信したドキュメントは要求ドキュメントと見なされていました。ただし、XAPI により、ERP 発信システムで要求ドキュメントを生成し、ERP 実行者システムに送信できます。応答ドキュメントは ERP 実行者システムで生成されて送信され、ERP 発信システムが受信します。XAPI をサポートし、XML ディスパッチ・カーネルで応答と返信を区別できるようにするために、J.D. Edwards は jdeResponse 要素と共に使用する次の新しい「Type」属性を作成しました。

jdeResponse=RealTimeEvent	この要素と属性は、ERP 発信システムからの XAPI 要求を識別し、ERP 実行者システムに送信するために使用します。
jdeResponse=xapicallmethod	この要素と属性は、ERP 実行者システムからの XAPI 応答を識別し、ERP 発信システムに送信するために使用します。

XMLDispatch カーネルは、要素と上記の「Type」属性の 1 つを含むドキュメントを受け取ると、そのドキュメントを XMLService カーネルに送ります。XMLService は、jdeResponse 要素に関連付けられた「Type」属性に基づいて応答か返信かを区別して、そのドキュメントを適切に処理できます。

XAPI の ERP 入力者および ERP 実行者のセキュリティ

ERP 入力者システムと ERP 実行者システムへのアクセスは、次の情報に基づいています。

- ユーザーID
- パスワード
- 環境
- ロール

ERP 発信システムは、上記のセキュリティ情報が有効かどうかを確認し、暗号化されたパスワードを使用して Huser オブジェクトを作成し、ERP 実行者システムに送信します。パスワードの暗号化とデコードには、暗号化 API(jdeEnchyper/jdeDecypher)が使用されます。セキュリティ情報は、XAPI 要求 XML ドキュメントの形式で送信されます。

注:

ユーザーID、パスワード、環境、およびロールは、両方の ERP システム(ERP 入力者と ERP 実行者)の間で一致する必要があります。

XAPI の ERP 入力者および ERP 実行者のエラー処理

2 つの ERP システム間での XAPI エラー処理中には、次の 2 種類のエラーが発生する場合があります。

- ビジネス関連エラー

  ビジネス関数またはビジネス関数スペックが見つかりません。

- システム・エラー

  この種のエラーは、メッセージ配信障害など、システムの他の部分で発生します。

ビジネス関連エラーに対する XAPI エラー処理は、次のように実行されます。

- XAPI システムが、ビジネス関連エラーをエンタープライズ・サーバーのログに記録します。エラーは XAPI 返信の一部として配信されます。
- XAPI API が、応答ドキュメントからのビジネス・エラーを解析します。
- XAPI は、エラーに関して使用可能なすべての情報をエンタープライズ・サーバーのログに記録します。

イベントの jde.ini ファイルの構成

エンタープライズ・サーバーの jde.ini ファイルは、Z イベント、リアルタイム・イベントおよび XAPI イベントの生成をサポートするように正しく構成する必要があります。テキスト・エディタを使用して、エンタープライズ・サーバーの jde.init ファイルの特定の設定を手動で編集し、検証してください。

注:

企業内に複数の ERP エンタープライズ・サーバーがある場合は、ロジック、バッチ、およびインタオペラビリティのすべてのセクションで、各サーバーに同じ設定をしておく必要があります。

次のカーネルと[JDEITDRV]の設定を使用して、エンタープライズ・サーバー上で jde.ini ファイルを構成します。生成したいイベントのタイプ(Z、リアルタイム、または XAPI)に該当するカーネルを構成してください。

注:

設定する必要のあるカーネルの判断と、イベント・タイプごとのその他の jde.ini 設定については、『インタオペラビリティ』ガイドの該当する「jde.ini ファイルの構成」を参照してください。

```
[JDENET_KERNEL_DEF19]
    krnlName=EVN KERNEL
    dispatchDLLName=jdeie.dll
    dispatchDLLFunction=_JDEK_DispatchITMessage@28
    maxNumberOfProcesses=1
    numberOfAutoStartProcesses=0

[JDENET_KERNEL_DEF20]
    krnlName=IEO KERNEL
    dispatchDLLName=jdeieo.dll
    dispatchDLLFunction=_JDEK_DispatchIEOMessage@28
    maxNumberOfProcesses=1
    numberOfAutoStartProcesses=0

[JDENET_KERNEL_DEF22]
    krnlName=XML Dispatch KERNEL
    dispatchDLLName=xmldispatch.dll
    dispatchDLLFunction=_XMLDispatch@28
    maxNumberOfProcesses=1
    numberOfAutoStartProcesses=0

[JDENET_KERNEL_DEF24]
    krnlName=XML Service KERNEL
    dispatchDLLName=xmlservice.dll
    dispatchDLLFunction=_XMLServiceDispatch@28
    maxNumberOfProcesses=1
    numberOfAutoStartProcesses=0
```

[JDEITDRV]
DrvCount=5
Drv1=Z:zdrv.dll
Drv2=RT:rtdrv.dll
Drv3=JDENET:jdetrdrv.dll
Drv4=MSMQ:msmqrtdrv.dll
Drv5=MQS:mqsrtdrv.dll

注:

jde.ini ファイルの[JDEITDRV]セクションで、イベント生成ドライバとトランスポート・ドライバを設定します。これらのドライバをすべて設定する必要はありません。たとえば、メッセージング・アダプタを使用しなければ、MSMQ および MQS 設定は不要です。使用中の設定の数で DrvCount を定義してください。

[JDENET]
MaxKernelRanges=27

注:

MaxKernelRanges 設定では、JDENET カーネルの最大数を指定します。この値は、定義したカーネルの合計数を含むように設定する必要があります。

上記の設定は Windows 2000 と Windows NT の場合です。次の表に、その他のプラットフォーム用の設定を示します。

	AS400	HP9000B	Sun または RS6000
EVN (19) dispatchDLLName	JDEIE	libjdeie.sl	libjdeie.so
EVN (19) dispatchDLLFunction	JDEK_DispatchITMessage	JDEK_DispatchITMessage	JDEK_DispatchITMessage
IEO (20) dispatchDLLName	JDEIEO	libjdeieo.sl	libjdeieo.so
IEO (20) dispatchDLLFunction	JDEK_DispatchIEOMessage	JDEK_DispatchIEOMessage	JDEK_DispatchIEOMessage
XML Dispatch (22) dispatchDLLName	XMLDSPATCH	libxmldispatch.sl	libxmldispatch.so
XML Dispatch (22) dispatchDLLFunction	JDEK_XMLDispatch	JDEK_XMLDispatch	JDEK_XMLDispatch
XML Service (24) dispatchDLLName	XMLSERVICE	libxmlservice.sl	libxmlservice.so
XML Service (24) dispatchDLLFunction	JDEK_XMLServiceDispatch	JDEK_XMLServiceDispatch	JDEK_XMLServiceDispatch
Drv1	RTDRV	librtdrv.sl	librtdrv.so
Drv2	ZDRV	libzdrv.sl	libzdrv.so
Drv3	JDETRDRV	libjdetrdrv.sl	libjdstrdrv.so
Drv4	MSMQRTDRV	libmsmqrtdrv.sl	libmsmqrtdrv.so
Drv5	MQSRTDRV	libmqsrtdrv.sl	libmqsrtdrv.so

MQSeries と MSMQ の jde.ini ファイルの構成

MQSeries または MSMQ を使用して ERP イベントを受信する場合は、jde.ini ファイル内で追加設定を定義する必要があります。

注:

MQSeries/MSMQ 送信待ち行列からのイベント・サブスクリプションでは、jde.ini ファイルの MQS または MSMQ セクションに指定されている ERP ユーザーID、パスワード、および環境が使用されます。

MQSeries を使用する場合は、[MQ SI]セクションで次の設定を定義してください。

[MQ SI]

設定	目的
QMGRName=	待ち行列マネージャ名。
QOutboundName=	MQSeries トランスポート・ドライバが XML メッセージを入れる待ち行列の名前。
eventType_QMGR=	特定の eventType 用の待ち行列マネージャの名前。たとえば、eventType が RTSOOUT の場合、設定は次のようになります。 RTSOOUT_QMGR=
eventType_Q=	特定の eventType 用の待ち行列の名前。たとえば、eventType が RTSOOUT の場合、設定は次のようになります。 RTSOOUT_Q=

MSMQ を使用する場合は、[MSMQ]セクションで次の設定を定義してください。

[MSMQ]

設定	目的
QLabelName=	待ち行列ラベル名。
QOutboundName=	MSMQ トランスポート・ドライバが XML メッセージを入れる待ち行列の名前。
eventType_Label=	特定の eventType 用の待ち行列ラベルの名前。たとえば、eventType が RTSOOUT の場合、設定は次のようになります。 RTSOOUT_Label=
eventType_Q=	特定の eventType 用の待ち行列の名前。たとえば、eventType が RTSOOUT の場合、設定は次のようになります。 RTSOOUT_Q=

jde.ini ファイルの[JDEITDRV]セクションで、イベント生成ドライバとトランスポート・ドライバを設定します。使用するトランスポート・ドライバのみを構成し、DrvCount には構成する設定の数と一致する値を入力してください。次に、[JDEITDRV]の設定例を示します。

```
[JDEITDRV]
DrvCount=5
Drv1=Z:zdrv.dll
```

Drv2=RT:rtdrv.dll
Drv3=JDENET:jdetrdrv.dll
Drv4=MSMQ:msmqrtdrv.dll
Drv5 =MQS:mqsrtdrv.dll

イベントの定義

新規の単一イベントおよびコンテナ・イベントを追加したり、既存のイベントを見直すには、〈イベント要求の定義〉プログラム(P90701)を使用します。イベント名ごとに単一イベントを追加します。単一イベントを追加するときは、データ構造体を指定する必要があります。コンテナ・イベントは、単一イベントまたは集合体イベントあるいはその両方を含んでいます。新規のコンテナ・イベントを追加するときは、イベント(個別に使用する単一イベント)またはデータ構造体(集合体にまとめる単一イベント)を定義します。単一イベントおよびコンテナ・イベントの情報は変更できます。単一イベントおよびコンテナ・イベントを削除することもできます。イベントの状況をアクティブまたは非アクティブに変更できます。システムに複数の環境が存在する場合、イベントの状況はすべての環境で同じになります。Z ファイル・イベントを定義するには、〈イベント定義ワークベンチ〉フォームで、[フォーム]メニューから[Z ファイル・イベント]を選びます。メニュー・オプションを使用してサブスクライバ情報にアクセスできます。

イベントが生成されたとき、IEO カーネルは、インタオペラビリティ・イベント記述テーブル(F90701)を読み取ってそのイベントについて調べます。指定されたイベントのカテゴリがデータベースで構成されたイベント・カテゴリと異なる場合、システムは次のようにエラーをログ・ファイルに書き込みます。

- データベースでイベントの定義が見つからない場合、次のエラー・メッセージが EVN ログに書き込まれます。

  Error in getNextNumberF0002(getNextNumberF0002 のエラー) : failed to open F0002 table.(F0002 テーブルを開けませんでした。)

  警告 : table F90702 doesn't exist.(テーブル F90702 がありません。)Some features will be turned off.(いくつかの機能が無効になります。)

- データベース・テーブルがない場合、次のメッセージが EVN ログに書き込まれます。

  CheckTableExists failed(CheckTableExists の失敗):invalid hEnv or hUser.(hEnv または hUser が無効です。)

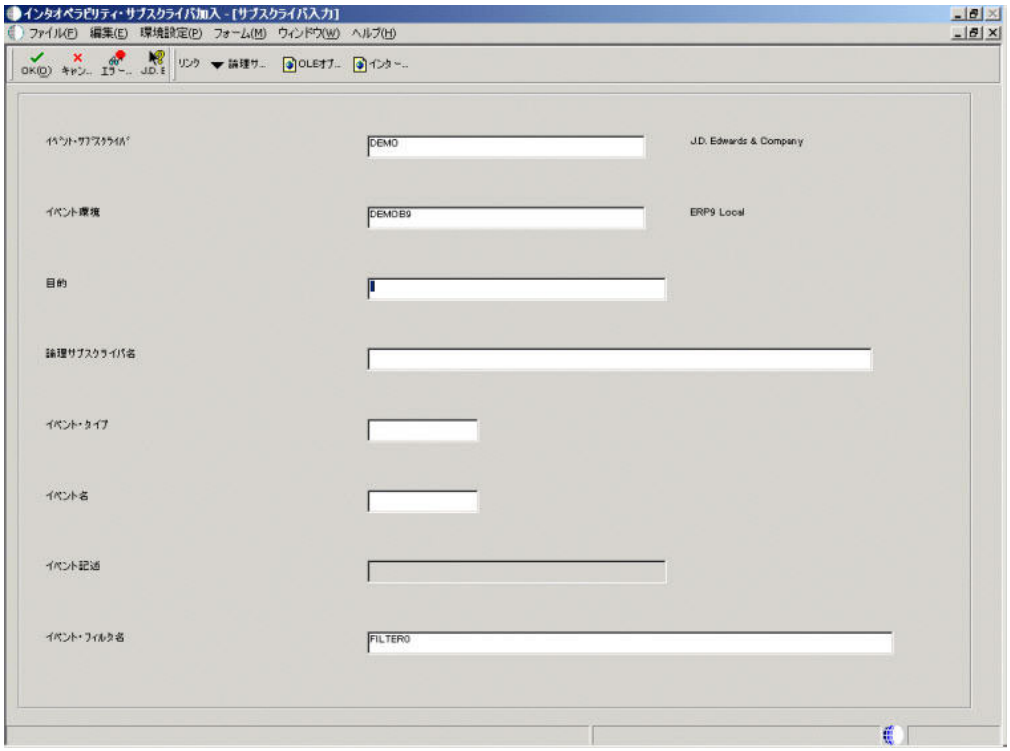
- また、次のメッセージが IEO ログに書き込まれます。

  警告 : table %s doesn't exist.(警告 : テーブル%s がありません。)Some features will be turned off.(いくつかの機能が無効になります。)

▶ 単一イベントを追加するには

〈インタオペラビリティ〉メニュー(GH9070)で、〈インタオペラビリティ・イベントの定義〉を選びます。または、[略式コマンド]に"P90701"と入力して〈イベント定義ワークベンチ〉フォームにアクセスします。

1. 〈イベント定義ワークベンチ〉で、[追加]をクリックします。



2. 〈イベント入力〉フォームで、次のフィールドに値を入力します。

   - イベント名
   - イベント記述
   - イベント・タイプ
   - イベント・カテゴリ
   - システム・コード
   - 安定した配信
   - タイムアウトしきい値
   - データ構造体

注:

高信頼イベント配信機能を使用する場合は、[安定した配信]フィールドを高信頼(1 または Y)に設定し、[タイムアウトしきい値]フィールドを設定する必要があります。[タイムアウトしきい値]フィールドの値は秒単位で、初期配信試行が失敗する高信頼イベントにのみ適用されます。このフィールドにより、イベントが作成されてから正常に配信できない場合に破棄されるまでの最大経過期間が確定されます。しきい値がゼロのイベントは期限が切れません。

---

3. ［OK］をクリックして、更新内容を保存します。

注：

XAPI イベントを追加すると、［イベント・カテゴリ］フィールドに"Container"が自動的に入力され、［OK］をクリックすると〈イベント定義の詳細〉フォームが表示されます。［データ構造体］および［データ記述］フィールドに値を入力して［OK］をクリックします。

4. ［キャンセル］をクリックして、〈イベント定義ワークベンチ〉に戻ります。
5. 〈イベント定義ワークベンチ〉で［検索］をクリックして、イベントを表示させます。
6. ［閉じる］をクリックしてメイン・メニューに戻ります。

フィールド記述

記述	用語解説
イベント名	イベント名。たとえば、JDERTSOOUT などがあります。単一イベントは別のイベントの一部です。
イベント記述	イベントの記述。
イベント・タイプ	イベント・タイプの名前を表す値。たとえば、RTE は Real Time Event を、ZFILE は Batch Upload Event を表します。
イベント・カテゴリ	イベントのカテゴリ。単一イベントまたはイベント・コンテナなどがあります。
システム・コード	システム・コードを示すユーザー定義コード(98/SY)
安定した配信	完了しなかったイベントを再送信して保管するかどうかを指定するオプション。このオプションをオフにすると、完了しなかったイベントは再送信および保管されません。このオプションをオンにすると、追加プロセスによりシステムのパフォーマンスに影響があることがあります。有効な値は次のとおりです。 1 または Y 完了しなかったイベントを再送信および保管する 0 または N 完了しなかったイベントを再送信および保管しない
タイムアウトしきい値	イベントが破棄された後の秒数。しきい値がゼロのイベントは期限が切れません。
データ構造体	イベント情報を渡すデータ構造体の名前。

► コンテナ・イベントを追加するには

〈インタオペラビリティ〉メニュー(GH9070)で、〈インタオペラビリティ・イベントの定義〉を選びます。または、［略式コマンド］に"P90701"と入力して〈イベント定義ワークベンチ〉フォームにアクセスします。

1. 〈イベント定義ワークベンチ〉で、［追加］をクリックします。
2. 〈イベント入力〉フォームで、次のフィールドに値を入力します。
   - イベント名

- イベント記述
- イベント・タイプ
- イベント・カテゴリ
- システム・コード
- 安定した配信
- タイムアウトしきい値

注:

高信頼イベント配信機能を使用する場合は、[安定した配信]フィールドを高信頼(1 または Y)に設定し、[タイムアウトしきい値]フィールドを設定する必要があります。[タイムアウトしきい値]フィールドの値は秒単位で、初期配信試行が失敗する高信頼イベントにのみ適用されます。このフィールドにより、イベントが作成されてから正常に配信できない場合に破棄されるまでの最大経過期間が確定されます。しきい値がゼロのイベントは期限が切れません。

3. [OK]をクリックして、〈イベント定義詳細〉にアクセスします。
4. 〈イベント定義詳細〉で次のいずれかを選択します。
   - イベント・データ
   - データ構造体データ

   注:
   コンテナ・イベントに対する個々の単一(複合)イベントを定義する場合は、[イベント・データ]を選びます。コンテナ・イベントに対する集合体イベントを定義する場合は、[データ構造体データ]を選びます。

5. 次のいずれかを実行します。
   - [イベント・データ]を選択した場合:
     - グリッド内の[単一イベント]フィールドの最初の空行で、ビジュアル・アシストをクリックします。
     - 〈イベントの検索/選択〉でイベントを選択してから、[選択]をクリックします。
   - [データ構造体データ]を選択した場合:
     - グリッドの[データ構造体]フィールド内の最初の空行で、ビジュアル・アシストをクリックします。
     - 〈個別オブジェクトの検索/選択〉でデータ構造体を選択してから、[選択]をクリックします。

     適切な単一イベントをコンテナ・イベントにリンクするのに必要な回数だけ、このステップを繰り返します。
6. 変更を保存して〈イベント入力〉に戻るには、[OK]をクリックします。

7. 〈イベント入力〉で[キャンセル]をクリックして、〈イベント定義ワークベンチ〉に戻ります。
8. 〈イベント定義ワークベンチ〉で[検索]をクリックして、イベントを表示させます。
9. [閉じる]をクリックしてメイン・メニューに戻ります。

► 単一イベントの定義を表示または変更するには

〈インタオペラビリティ〉メニュー(GH9070)で、〈インタオペラビリティ・イベント定義〉を選びます。または、[略式コマンド]に"P90701"と入力して〈イベント定義ワークベンチ〉フォームにアクセスします。

1. 〈イベント定義ワークベンチ〉フォームで、次のフィールドに値を入力します。
   - イベント名
   - 記述
   - イベント・タイプ
   - システムコード
2. 次のオプションのいずれかをクリックしてから、[検索]をクリックして既存のイベントを表示します。
   - アクティブな状況
   - 非アクティブな状況
   - すべての状況
3. 単一イベントを選んで[選択]をクリックします。
4. 〈イベント入力〉フォームで、次のフィールドを表示させるか、または変更します。
   - イベント記述
   - イベント・タイプ
   - システム・コード
   - 安定した配信
   - データ構造体
5. [OK]をクリックして、変更を保存し、〈イベント定義ワークベンチ〉に戻ります。
6. 〈イベント定義ワークベンチ〉で[閉じる]をクリックして、メイン・メニューに戻ります。

▶ 単一イベントをコンテナ・イベントに変更するには

〈インタオペラビリティ〉メニュー(GH9070)で、〈インタオペラビリティ・イベント定義〉を選びます。または、[略式コマンド]に"P90701"と入力して〈イベント定義ワークベンチ〉フォームにアクセスします。

1. 〈イベント定義ワークベンチ〉フォームで、次のフィールドに値を入力します。
   - イベント名
   - 記述
   - イベント・タイプ
   - システムコード
2. 次のオプションのいずれかをクリックしてから、[検索]をクリックして既存のイベントを表示します。
   - アクティブな状況
   - 非アクティブな状況
   - すべての状況
3. コンテナ・イベントに変更する単一イベントを選択してから、[選択]をクリックします。
4. 〈イベント入力〉フォームで、次のフィールドを変更します。
   - イベント・カテゴリ
5. [OK]をクリックして、〈イベント定義詳細〉にアクセスします。
6. 〈イベント定義詳細〉において、グリッド内の[単一イベント]フィールドの最初の空行で、ビジュアル・アシストをクリックします。
7. 〈イベントの検索/選択〉でイベントを選択してから、[選択]をクリックします。

   適切な単一イベントをコンテナ・イベントにリンクするのに必要な回数だけ、ステップ 6 および 7 を繰り返します。
8. [OK]をクリックして、変更を保存し、〈イベント定義ワークベンチ〉に戻ります。
9. 〈イベント定義ワークベンチ〉で[検索]をクリックして、イベントを表示させます。
10. [閉じる]をクリックしてメイン・メニューに戻ります。

► コンテナ･イベントの定義を表示または変更するには

〈インタオペラビリティ〉メニュー(GH9070)で、〈インタオペラビリティ・イベント定義〉を選びます。または、[略式コマンド]に"P90701"と入力して〈イベント定義ワークベンチ〉フォームにアクセスします。

1. 〈イベント定義ワークベンチ〉フォームで、次のフィールドに値を入力します。
    - イベント名
    - 記述
    - イベント・タイプ
    - システムコード
2. 次のオプションのいずれかをクリックしてから、[検索]をクリックして既存のイベントを表示します。
    - アクティブな状況
    - 非アクティブな状況
    - すべての状況
3. 表示または変更するイベントを含む明細行を選択してから、[選択]をクリックします。
4. 〈イベント入力〉フォームで、次のフィールドを表示させるか、または変更します。
    - イベント記述
    - イベント・タイプ
    - システム・コード
    - 安定した配信
5. [OK]をクリックして、すべての変更を保存し、〈イベント定義詳細〉にアクセスします。
6. 〈イベント定義詳細〉で次のいずれかを実行します。
    - コンテナ･イベントに関連する単一イベントを表示させます。
    - 単一イベントをコンテナ･イベントに追加します。
        - グリッド内で[単一イベント]フィールドの中の空のローでビジュアル･アシストをクリックします。
        - ビジュアル･アシストをクリックすると、〈イベントの検索/選択〉フォームが表示されます。イベント･タイプを選択して[選択]をクリックします。コンテナ･イベントにリンクする各単一イベント・タイプごとに、この処理を繰り返します。
    - コンテナ･イベントに関連する既存の単一イベントを変更してください。
        - グリッド内で、変更するイベントを含む明細行を選択してから、[単一イベント]フィールドに表示されるビジュアル･アシストをクリックします。

- ビジュアル･アシストをクリックすると、〈イベントの検索/選択〉フォームが表示されます。イベント・タイプを選択してから、[選択]をクリックして〈イベントの定義詳細〉フォームに戻ります。選択したローが更新されます。

- 単一イベントをコンテナ･イベントから削除してください。
  - グリッド内で、削除するイベントを含む明細行を選択してから、[削除]をクリックします。
  - 削除を確認します。

7. [OK]をクリックして、変更を保存し、〈イベント定義ワークベンチ〉に戻ります。
8. 〈イベント定義ワークベンチ〉で[キャンセル]をクリックして、メインメニューに戻ります。

▶ コンテナ･イベントを単一イベントに変更するには

〈インタオペラビリティ〉メニュー(GH9070)で、〈インタオペラビリティ・イベント定義〉を選びます。または、[略式コマンド]に"P90701"と入力して〈イベント定義ワークベンチ〉フォームにアクセスします。

1. 〈イベント定義ワークベンチ〉フォームで、次のフィールドに値を入力します。
   - イベント名
   - 記述
   - イベント・タイプ
   - システムコード
2. 次のオプションのいずれかをクリックしてから、[検索]をクリックして既存のイベントを表示します。
   - アクティブな状況
   - 非アクティブな状況
   - すべての状況
3. 単一イベントに変更するコンテナ･イベントを選択してから、[選択]をクリックします。
4. 〈イベント入力〉フォームで、次のフィールドを変更します。
   - イベント・カテゴリ
5. [OK]をクリックします。

   変更によって詳細が失われることを示す警告メッセージが表示されます。
6. 警告メッセージで[OK]をクリックします。
7. [データ構造]フィールドに値を入力するには、ビジュアル･アシストをクリックします。
8. 〈個別オブジェクトの検索/選択〉でデータ構造体を選択してから、[選択]をクリックします。
9. 〈イベント定義詳細〉で[OK]をクリックして、変更を保存し、〈イベント定義ワークベンチ〉に戻ります。
10. 〈イベント定義ワークベンチ〉で[キャンセル]をクリックして、メイン・メニューに戻ります。

► イベントの状況を変更するには

〈インタオペラビリティ〉メニュー(GH9070)から、〈インタオペラビリティ・イベント定義〉を選択します。その他に、略式コマンド行で P90701 と入力して、〈イベント定義ワークベンチ〉フォームにアクセスする方法もあります。

1. 〈イベント定義ワークベンチ〉フォームで、次のフィールドに値を入力します。
    - イベント名
    - 記述
    - イベント・タイプ
    - システムコード
2. 次のオプションをクリックしてから、[検索]をクリックして既存のイベントを表示します。
    - すべての状況
3. グリッド内で、状況を変更したいイベントを選択します。
4. [ロー]メニューから[変更状況]を選びます。
5. 状況の変更を表示させるには、[検索]をクリックします。
6. [閉じる]をクリックしてメイン・メニューに戻ります。

注:

イベントの状況は、どの環境でも同じです。あるイベントがアクティブな場合、そのイベントは全環境でアクティブであり、非アクティブな場合は全環境でアクティブです。

► イベントをコピーするには

〈インタオペラビリティ〉メニュー(GH9070)で、〈インタオペラビリティ・イベント定義〉を選びます。または、[略式コマンド]に"P90701"と入力して〈イベント定義ワークベンチ〉フォームにアクセスします。

1. 〈イベント定義ワークベンチ〉フォームで、次のフィールドに値を入力します。
    - イベント名
    - 記述
    - イベント・タイプ
    - システムコード
2. 次のオプションのいずれかをオンにしてから、[検索]をクリックします。
    - アクティブな状況

- 非アクティブな状況
- すべての状況

3. コピーするイベントを含む明細行を選択してから、[コピー]をクリックします。
4. 〈イベント入力〉フォームで次のフィールドに値を入力します。
   - イベント名
5. 次のいずれかを実行します。
   - 単一イベントをコピーした場合は、新規イベントの定義を修正します。

     注:
     『インタオペラビリティ』ガイドの「単一イベントの定義を表示または変更するには」のステップ 4～6 を参照してください。

   - コンテナ・イベントをコピーした場合は、新規イベントの定義を修正します。

     注:
     『インタオペラビリティ』ガイドの「コンテナ・イベントの定義を表示または変更するには」のステップ 4～8 を参照してください。

► イベントを削除するには

〈インタオペラビリティ〉メニュー(GH9070)で、〈インタオペラビリティ・イベント定義〉を選びます。または、[略式コマンド]に"P90701"と入力して〈イベント定義ワークベンチ〉フォームにアクセスします。

1. 〈イベント定義ワークベンチ〉フォームで、次のフィールドに値を入力します。
   - イベント名
   - 記述
   - イベント・タイプ
   - システムコード
2. 次のオプションをオンにして、[検索]をクリックします。
   - アクティブな状況
3. 削除するイベントを含む明細行を選択してから、[削除]をクリックします。

   コンテナ・イベントを削除する場合は、削除されたことを確認する必要があります。
4. イベントが削除されたことを検証するには、[検索]をクリックします。
5. [閉じる]をクリックしてメイン・メニューに戻ります。

サブスクライバ情報の設定

XAPI イベントの場合は、それに対する応答を受信できるように、インタオペラビリティ・サブスクライバ加入テーブル(F90702)を更新する必要があります。各 XAPI イベントには論理サブスクライバが必要ですが、その設定が必要な場合があります。Z イベントとリアルタイム・イベントの場合は、イベントの作成時にこのテーブルが自動的に更新されます。このテーブルを使用すると、Z イベントとリアルタイム・イベントに対する持続的なサブスクリプションを表示できます。新しいサブスクリプション情報を追加したり、既存のサブスクリプション情報の検討と変更を行うには、〈インタオペラビリティ・イベント・サブスクリプション〉プログラム(P90702)を使用します。また、既存のサブスクリプションをコピーし修正することによって、新規のサブスクリプションを追加することも、サブスクリプションを削除することも可能です。

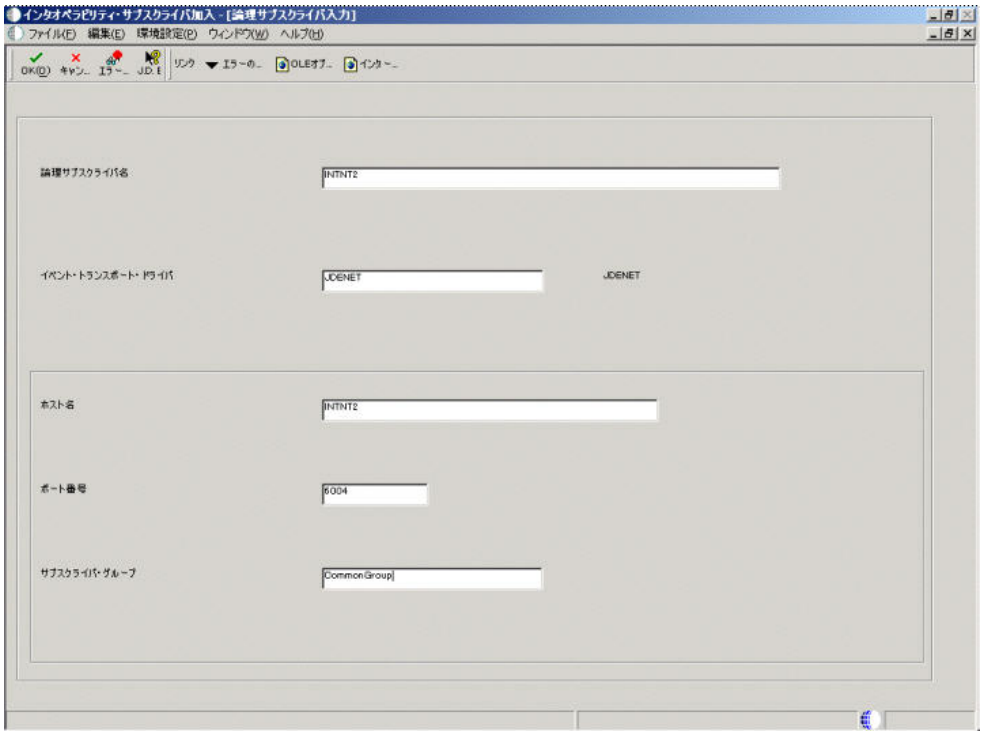
テーブル(F90702)には、マシン名やポート番号などのサブスクライバ情報が含まれており、EVN がこれを読み出します。イベントのサブスクライバ情報がない場合も、XAPI イベントは生成されますが、送信は行われません。論理サブスクライバを設定するには、[フォーム]メニューから[論理サブスクライバ]を選びます。

[フォーム]メニューから[イベント定義]を選ぶと、リアルタイム・イベントと XAPI イベントの定義にアクセスして表示できます。さらに、〈サブスクライバ・ワークベンチ〉フォームの[Z ファイル・イベント]アイコンをクリックするか、または[フォーム]メニューの[Z ファイル・イベント]オプションを選んで、Z イベントにアクセスし、表示させることができます。

► 論理サブスクライバを追加するには

〈インタオペラビリティ〉メニュー(GH9070)で、〈インタオペラビリティ・サブスクライバ加入〉を選びます。または、[略式コマンド]に"P90702"と入力して〈サブスクライバ・ワークベンチ〉フォームにアクセスします。

1. 〈サブスクライバ・ワークベンチ〉で、[フォーム]メニューから[論理サブスクライバ]を選びます。
2. 〈論理サブスクライバ名の処理〉で、[追加]をクリックします。



3. 〈論理サブスクライバ入力〉フォームで、次のフィールドに値を入力します。
   - 論理サブスクライバ名

     このフィールドではスペースを使用しないでください。
   - イベント・トランスポート・ドライバ
   - ホスト名
   - ポート番号
4. 次のフィールドに"CommonGroup"を入力します。
   - サブスクライバ・グループ
5. [OK]をクリックして、更新内容を保存します。
6. [キャンセル]をクリックして〈論理サブスクライバ名の処理〉に戻ります。
7. [閉じる]をクリックして、〈サブスクライバ・ワークベンチ〉に戻ります。

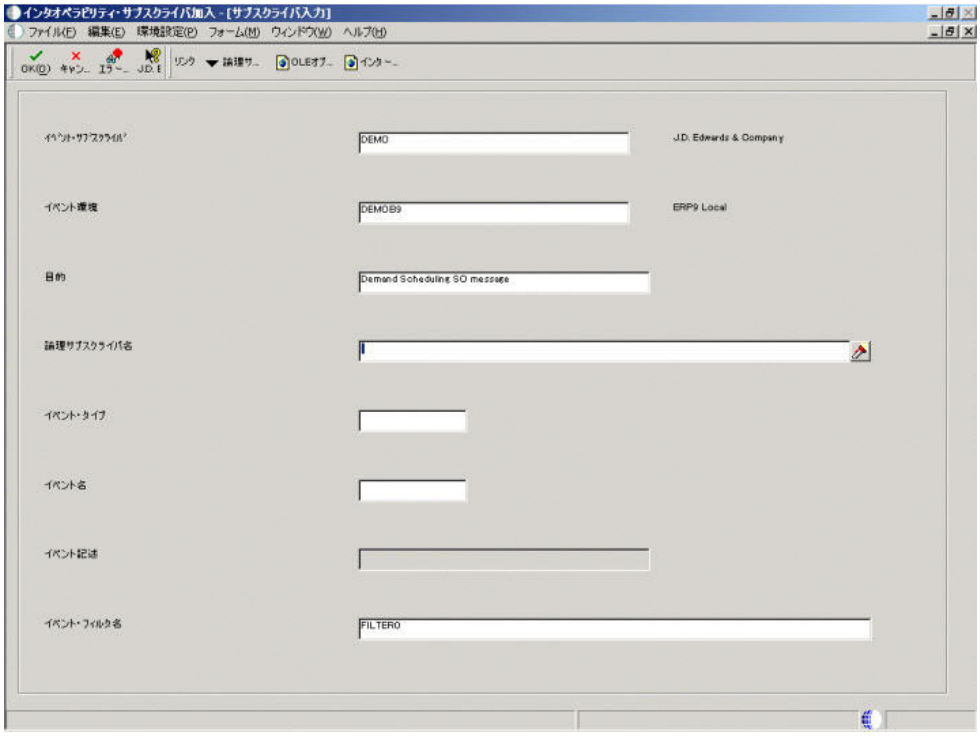
フィールド記述

記述	用語解説
論理サブスクライバ名	サブスクライバを識別する値。
イベント・トランスポート・ドライバ	イベントを JDENET などのサブスクライバに配信するトランスポート・ドライバ名。
ホスト名	サブスクライバのイベントを処理するサーバー名。
ポート番号	サブスクライバ・サービスが実行されれいるポートを識別する番号。
サブスクライバ・グループ	サブスクライバのイベントを背信する方法を指定するユーザー定義名。たとえば XPI アダプタを使用している場合は、アダプタ名を入力してください。

▶ サブスクリプションを追加するには

〈インタオペラビリティ〉メニュー(GH9070)で、〈インタオペラビリティ・サブスクライバ・エンロールメント〉を選びます。または、[略式コマンド]に"P90702"と入力して〈サブスクライバ・ワークベンチ〉フォームにアクセスします。

1. 〈サブスクライバ・ワークベンチ〉で、[追加]をクリックします。



2. 〈サブスクライバ入力〉フォームで、次のフィールドに値を入力します。

   - イベントサブスクライバ
   - イベント環境
   - 目的
   - 論理サブスクライバ名
   - イベント・タイプ

3. 次のフィールドには"Filter0"が自動的に入力されます。

   - イベント名

4. [OK]をクリックして、更新内容を保存します。
5. [キャンセル]をクリックして、〈サブスクライバ・ワークベンチ〉に戻ります。
6. 〈サブスクライバ・ワークベンチ〉で[検索]をクリックして、サブスクリプションを表示させます。

7. [閉じる]をクリックしてメイン・メニューに戻ります。

フィールド記述

記述	用語解説
イベントサブスクライバ	サブスクライバのユーザーID。
イベント環境	イベントが実行される環境を識別する値。
目的	ユーザー定義名称または備考。
論理サブスクライバ名	サブスクライバを識別する値。
イベント・タイプ	イベント・タイプの名前を表す値。たとえば、RTE は Real Time Event を、ZFILE は Batch Upload Event を表します。
イベント名	イベント名。たとえば、JDERTSOOUT などがあります。単一イベントは別のイベントの一部です。
イベント・フィルタ名	XML トランスフォーメーションで使用する XLST ルールを含むファイル名。Transform.xml などがあります。

► サブスクリプションを表示または変更するには

〈インタオペラビリティ〉メニュー(GH9070)で、〈インタオペラビリティ・サブスクライバ・エンロールメント〉を選びます。または、[略式コマンド]に"P90702"と入力して〈サブスクライバ・ワークベンチ〉フォームにアクセスします。

1. 〈サブスクライバ・ワークベンチ〉フォームで、次のフィールドに値を入力します。
   - サブスクライバ名
   - 目的
2. 次のオプションのいずれかをクリックしてから、[検索]をクリックして既存のサブスクリプションを表示します。
   - アクティブな状況
   - 非アクティブな状況
   - すべての状況
3. 表示または変更したいサブスクリプションを選択してから、[選択]をクリックします。
4. 〈サブスクライバの入力〉フォームで、次のフィールドのいずれかを表示させるか、または変更します。
   - 目的
5. [OK]をクリックして、更新内容を保存し、〈サブスクライバ・ワークベンチ〉に戻ります。
6. 〈サブスクライバ・ワークベンチ〉で[検索]をクリックして、サブスクリプションを表示させます。
7. [閉じる]をクリックしてメイン・メニューに戻ります。

フィールド記述

記述	用語解説
サブスクライバ名	サブスクライバのユーザーID。
アクティブな状況	サブスクリプション状況がアクティブまたは非アクティブかを示す値。
非アクティブな状況	サブスクリプション状況がアクティブまたは非アクティブかを示す値。
すべての状況	サブスクリプション状況がアクティブまたは非アクティブかを示す値。

► サブスクリプションの状況を変更するには

〈インタオペラビリティ〉メニュー(GH9070)で、〈インタオペラビリティ・サブスクライバ・エンロールメント〉を選びます。または、[略式コマンド]に"P90702"と入力して〈サブスクライバ・ワークベンチ〉フォームにアクセスします。

1. 〈サブスクライバ・ワークベンチ〉フォームで、次のフィールドに値を入力します。
   - サブスクライバ名
   - 目的
2. 次のオプションをオンにしてから、[検索]をクリックして既存のサブスクリプションを表示します。
   - すべての状況
3. グリッド内で、状況を変更したいイベントを選択します。
4. [ロー]メニューから[変更状況]を選びます。
5. 状況の変更を表示させるには、[検索]をクリックします。
6. [閉じる]をクリックしてメイン・メニューに戻ります。

► サブスクリプションをコピーするには

〈インタオペラビリティ〉メニュー(GH9070)で、〈インタオペラビリティ・サブスクライバ・エンロールメント〉を選びます。または、[略式コマンド]に"P90702"と入力して〈サブスクライバ・ワークベンチ〉フォームにアクセスします。

1. 〈サブスクライバ・ワークベンチ〉フォームで、次のフィールドに値を入力します。
   - サブスクライバ名
   - 目的
2. 次のオプションのいずれかをオンにしてから、[検索]をクリックして既存のサブスクリプションを表示します。
   - アクティブな状況

- 非アクティブな状況
- すべての状況

3. コピーするイベントを含む明細行を選択してから、[コピー]をクリックします。
4. 〈サブスクライバの入力〉フォームで、次のフィールドを更新します。
   - イベントサブスクライバ
   - イベント環境
   - 目的
   - 論理サブスクライバ名
   - イベント・タイプ
   - イベント名
5. [OK]をクリックして、〈サブスクライバ・ワークベンチ〉に戻ります。
6. 〈サブスクライバ・ワークベンチ〉で[検索]をクリックして、サブスクリプションを表示させます。
7. [閉じる]をクリックしてメイン・メニューに戻ります。

► サブスクリプションを削除するには

〈インタオペラビリティ〉メニュー(GH9070)で、〈インタオペラビリティ・サブスクライバ・エンロールメント〉を選びます。または、[略式コマンド]に"P90702"と入力して〈サブスクライバ・ワークベンチ〉フォームにアクセスします。

1. 〈サブスクライバ・ワークベンチ〉フォームで、次のフィールドに値を入力します。
   - サブスクライバ名
   - 目的
2. 次のオプションをオンにして、[検索]をクリックします。
   - すべての状況
3. 削除するサブスクリプションを含む明細行を選択してから、[削除]をクリックします。
4. イベントが削除されたことを検証するには、[検索]をクリックします。
5. [閉じる]をクリックしてメイン・メニューに戻ります。

高信頼イベント配信

高信頼イベント配信は、Z イベント、リアルタイム･イベント、および XAPI イベントをサポートしています。高信頼イベント配信機能を使用するには、イベントをデータベース･テーブル内で定義する必要があります。jde.ini ファイル内では定義できません。

JDENET トランスポートは、Z イベント、リアルタイム･イベント、および XAPI イベントを配信します。高信頼イベント配信では、ネットワークの障害など転送の問題が生じた場合にも、確実にイベントを回復し、配信します。次の例は、イベントが失われた可能性があっても、それを回復して配信することが可能な状況を表します。

- JDENET プロセスが停止しています。
- 送信側と受信側間のネットワーク･リンクが永続的に停止しているため、JDENET は配信に失敗しました。
- 受信側カーネルの IPC バッファがいっぱいになっているため、JDENET は配信に失敗しました（送信側と受信側は異なるボックスにあります）。

注意:

高信頼配信の対象となるのは、イベントのトランスポートに関連する障害のみです。「単発」タイプの保証は提供しません。プロセス障害（クライアントとサーバー）があると、イベントが消失して回復できない（または重複して配信される）場合があります。

リアルタイム･イベントの配信は、リアルタイム API と Java コネクタ（IEO カーネルおよび EVN カーネル）間の転送に障害があった場合にも、信頼性を保ちます。XAPI 送信イベントの配信は、XAPI API と Java コネクタ（IEO カーネルおよび EVN カーネル）間の転送に障害があった場合にも、信頼性を保ちます。Z イベントの配信は、Z イベント･ジェネレータと Java コネクタ間の転送に障害があった場合にも信頼性を保ちます。

イベントが高信頼型であるか揮発型であるかに基づいて、信頼性のレベルを構成できます。揮発型イベントは、ネットワークまたはプロセスに障害が起こった場合に失われる可能性のあるイベントで、配信の信頼性は低くなります。高信頼イベントが失われる可能性があるのは、プロセス障害が発生した場合のみです。すべてのイベント･タイプごとに信頼性のレベルを構成できます。信頼性のレベルは、そのイベントのビジネス上の重要性によって異なります。たとえば、照会については揮発型に構成します。照会はビジネス･イベントとしての重要度が低く、イベントが失われたかどうかシステムが頻繁にチェックするのは無駄が大きいためです。購買オーダーは高信頼型に構成します。購買オーダーは重要なビジネス･イベントであり、システムがイベントを頻繁にチェックして取引の更新を行うようにする必要があるためです。揮発型イベントは、高信頼型イベントよりも高速に処理できますが、イベントが転送の間に失われた場合、配信の信頼性は低くなります。

リアルタイム･イベントおよび XAPI イベントは、単一イベント、集合体イベント、または複合イベントの形態をとることができます。複合イベントは、単一イベントで構成されています。複合イベントと、複合イベントを構成する単一イベントは、信頼性レベルが異なっていてもかまいません。たとえば、複合イベントについては、信頼性レベルを高信頼型として、RTSOOUT として登録します。そして単一イベントについては、信頼性レベルを揮発型として、RTSOLINE として登録します。RTSOOUT で構成した信頼性レベルが、RTSOLINE で構成した信頼性レベルを一時変更することはありません。それは、イベントの信頼性がイベント･タイプに基づいて適用されるためです。単一イベントはそれほど重要でなく、高信頼配信として構成する必要はないと判断した場合、複合イベント作成の間に作成される単一イベントの信頼性レベルは、他の単一イベントと同じになります。

リアルタイム・イベントおよび XAPI イベントの作成に使用する API は、信頼性レベルに影響されません。

高信頼イベント配信用のシステムの構成

高信頼イベント配信機能を使用するには、イベントを単一およびコンテナ・イベント定義テーブル(F90701)内で定義する必要があります。このタスクには、〈インタオペラビリティ・イベント定義〉プログラム(P90701)を使用します。〈イベント入力〉フォームで、[高信頼配信]フィールドを高信頼(Y または 1)に設定し、[タイムアウトしきい値]フィールドを設定する必要があります。[タイムアウトしきい値]フィールドの値は秒単位で、初期配信試行が失敗する高信頼イベントにのみ適用されます。このフィールドにより、イベントが作成されてから正常に配信できない場合に破棄されるまでの最大経過期間が確定されます。しきい値がゼロのイベントは期限が切れません。

イベント・プロトコル・テーブル(F90704)およびイベント・リンク・テーブル(F90703)の 2 つのテーブルによって、送信側と受信側間の通信が可能になります。イベント・プロトコル・テーブルは、イベントを配信するプロトコルに関する情報を保存します。イベント・リンク・テーブルは、最初の配信が失敗した高信頼イベントに関する情報を保存します。イベントが作成されると、これらのテーブルはシステムによって更新されます。

注意:

送信側と受信側の両方が、インタオペラビリティ・データベース・テーブルの同じインスタンスにアクセスする必要があります(データ・ソースが同じ)。

注:

高信頼イベントが見つからない場合、クライアント、Callobject、IEO、および EVN のログに次のメッセージが生成される場合があります。

> RDEL0000045 – Could not open tables for reliable event delivery (F90703 and F90704).
> (高信頼イベント配信のテーブルがオープンできませんでした(F90703 および F90704)。)
> Reliable event delivery will be disabled.(高信頼イベント配信は無効になります。)

このエラー・メッセージが表示される場合は、イベントが単一およびコンテナ・イベント定義テーブル(F90701)で定義されていること、[高信頼配信]および[タイムアウトしきい値]フィールドが上記のように設定されていること、イベント・プロトコル・テーブルとイベント・リンク・テーブルが存在することを確認してください。

イベント自己診断ユーティリティ・ツール

イベント自己診断ユーティリティ・ツールは、Z イベントとリアルタイム・イベントをサポートしています。通常は、システム管理者がこのツールを実行して、イベント・インフラストラクチャ機能が正常に動作しているかどうかを確認します。この自己診断ユーティリティ・ツールは、次のプラットフォームで使用できます。

- Windows 2000 と Windows NT
- AS400
- HP
- Sun
- AIX

イベント自己診断ユーティリティー・ツールは、イベントのインフラストラクチャを分析し、システムでイベントを処理するときに検出された構成、カーネル、およびネットワークの問題を報告します。このツールを用いて、総合的な分析を実行できます。あるいは、特定のニーズに対応した分析を行うようにツールを構成することもできます。イベント自己診断ツールは、イベント情報内のデータの破損を検出するため XML 比較ユーティリティを用いて XML ドキュメントを比較します。また、問題解決のための対策を提案します。このツールは、サーバーまたはクライアント、あるいはその両方で実行できます。

イベント自己診断ユーティリティ・ツールの処理概要

サーバー上、またはクライアントのアプリケーション API で、呼出しオブジェクト API によってイベントが生成された後に、イベント処理の失敗の原因となる問題が発生することがあります。次のような問題が発生する可能性があります。

- jde.ini ファイルに構成エラーがある。
- ZEVG ライブラリが使用できないか、IEO または EVN カーネル・プロセスが停止している。
- サブスクライバおよびサポートされるイベントが正しくロードされていない。
- イベント配信に関与する 1 つ以上のカーネルが、イベント情報を破損している。
- このインフラストラクチャに関わるコンポーネントのいずれか、またはすべてのコンポーネント間のネットワーク・リンクが永続的に停止している。

イベント自己診断ツールは、問題を検出すると、問題の説明と解決方法の提案を含むメッセージをユーザーに送信します。また、適切なログ・ファイルにエラーを記録します。送信されるメッセージは、どのログ・ファイルを確認する必要があるかを示します。次のリストに、イベント自己診断ツールによる問題の検出方法の例を示します。

- jde.ini ファイルについてインタオペラビリティを中心に詳細な分析を実行する。
- インタオペラビリティ・イベント記述テーブル(F90701)を読み取って、イベントが定義されているかどうかを調べる。
- インタオペラビリティ・サブスクライバ加入テーブル(F90702)を読み取って、持続的なサブスクリプション/サブスクリプション解除要求が、ツールから EVN カーネルに正常に送信されたかどうかを調べる。

- 〈オブジェクト構成マネージャ〉を読み取って、IEO カーネルの位置を検索する。このプロセスでは、RTE オブジェクトに対してアクティブな入力が 1 つのみであることが確認されます。
- 自己診断接続性メッセージ・コールを送信して、イベント・インフラストラクチャ内部の相互接続性をチェックする。
- 自己診断イベントを生成して、インフラストラクチャが提供するさまざまなサービスをテストし、データの破損がないようにイベント情報を検証する。

注:

上記のリストは概略であり、すべての検出操作が含まれているわけではありません。

イベント自己診断ユーティリティ・ツールのコンポーネント

イベント自己診断ユーティリティ・ツールは、次の 3 つのコンポーネントで構成されています。

- イベント・ジェネレータ
- イベント・レシーバ
- XML 比較ユーティリティ

イベント・ジェネレータ

イベント自己診断ユーティリティ・ツールは、まずイベント・ジェネレータ処理から診断を開始します。起動中に、イベント・ジェネレータがイベント・インフラストラクチャの基本的な背景分析を行います。内容は次のとおりです。

- jde.ini ファイルのインタオペラビリティ固有のセクションの検証
- リアルタイム・イベント定義の検証
- イベント・インフラストラクチャ内部のコンポーネント間接続性のチェック

起動が正常に行われると、イベント・ジェネレータは、イベント・インフラストラクチャが提供するさまざまな機能をテストします。さまざまなイベント・タイプの生成とテスト、有効なイベントのリスト表示、イベント・テンプレートのチェック、およびサブスクリプション情報のテストなどの機能です。次の方法のいずれかを用いて、1 つ以上のテストを実行できます。

- 以前にセットアップした既存の設定ファイルに対してテストを実行する。
- これからセットアップする新規の設定ファイルに対してテストを実行する。
- ツールのコマンド行メニューからオプションを選んでテストを実行する。

自己診断イベント生成が完了すると、そのイベントはイベント・インフラストラクチャ・システムを通過します。システムを通じて伝達されているイベント情報の正確さをテストするため、イベント・ジェネレータは XML ストリームの形でイベントに追加パケットを付加します。診断 XML パケットには、イベントに関する情報が含まれています。通信のそれぞれの段階で、標準メッセージ・パケットと自己診断 XML パケットを比較することによって、各カーネル（またはコンポーネント）はイベント情報を検証します。カーネル（またはコンポーネント）は、各比較ポイントで、対応するログ・ファイルこの比較結果を記録します。テンプレート要求の情報の正確さについても、同様にしてテストします。

イベント・レシーバ

イベント･レシーバは、EVN カーネル起動時に自己診断イベントを自らサブスクライブする NULL トランスポート・ドライバとして機能します。イベント･レシーバは、受信した自己診断イベントに含まれる XML ドキュメントを比較、検証します。イベント･レシーバは、この比較結果を EVN カーネルのログ･ファイルに記録します。

XML 比較ユーティリティ

イベント･ジェネレータは、XML 比較ユーティリティ･ツールを使用して、システムを通過するイベント情報またはイベント･テンプレート要求の正確性をテストします。XML 比較ユーティリティは 2 つの XML ドキュメントを比較して、それらが等しいか、または類似しているか、あるいはその両方について調べます。XML 比較ユーティリティでは、比較を実行するとき 3 つの XML ドキュメントが必要になります。そのうちの 2 つのドキュメントは、比較する実際の XML ドキュメントです。3 番目のドキュメントは除外 XML ドキュメントで、2 つの XML ドキュメントを比較する際に無視するノードが記述されています。

自己診断イベント生成のデータベース･テーブルの設定

データベース･テーブルを用いてイベントを生成する場合、インタオペラビリティ･イベント定義テーブル(F90701)に自己診断イベントを含める必要があります。〈インタオペラビリティ･イベント定義〉プログラム(P90701)を使用して、コンテナ･イベントと単一イベントの両方を追加してください。

イベント自己診断ツールの実行

イベント自己診断ツールを使用するには、有効な ERP ユーザーID、パスワード、および環境を指定する必要があります。このツールを ERP サーバーから使用する場合は、この情報をパラメータとして指定しなければ、ユーザー名、パスワード、および環境情報はサーバーjde.ini ファイルの［SECURITY］セクションから読み取られます。クライアントを使用する場合は、有効な ERP ユーザー名、パスワード、および環境を入力する必要があります。この情報を入力しないと、ツールが停止します。クライアントからイベントを生成する場合は、RTE または Z イベントから有効なサーバーへの有効な OCM マッピングも指定する必要があります。

はじめる前に

- システム上で PORTTEST が正常に実行されることを確認します。
- IEO および EVN カーネルの単一インスタンスが実行中であることを確認します。

ツールの起動

エンタープライズ･サーバーでイベント自己診断ツールを起動するには、次の場所で実行可能ファイルをダブルクリックします。

```
$system¥bin32¥sdtool.exe
```

または次のようにパラメータを指定する方法もあります。

```
$system¥bin32¥sdtool.exe username password environment
```

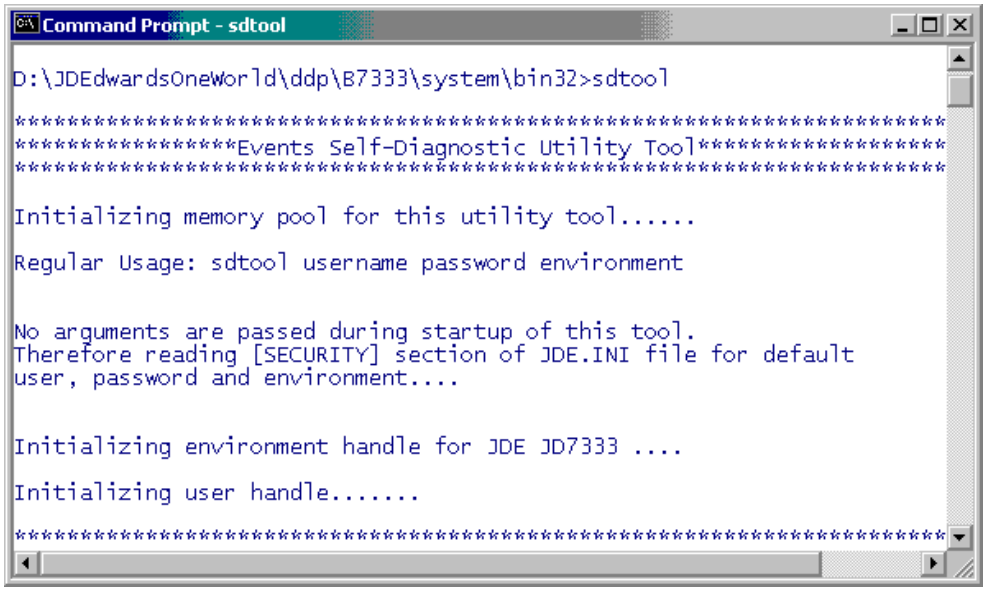
クライアント側からツールを起動するには、次のようにパスワードを指定する必要があります。

$system¥bin32¥sdtool.exe username password environment

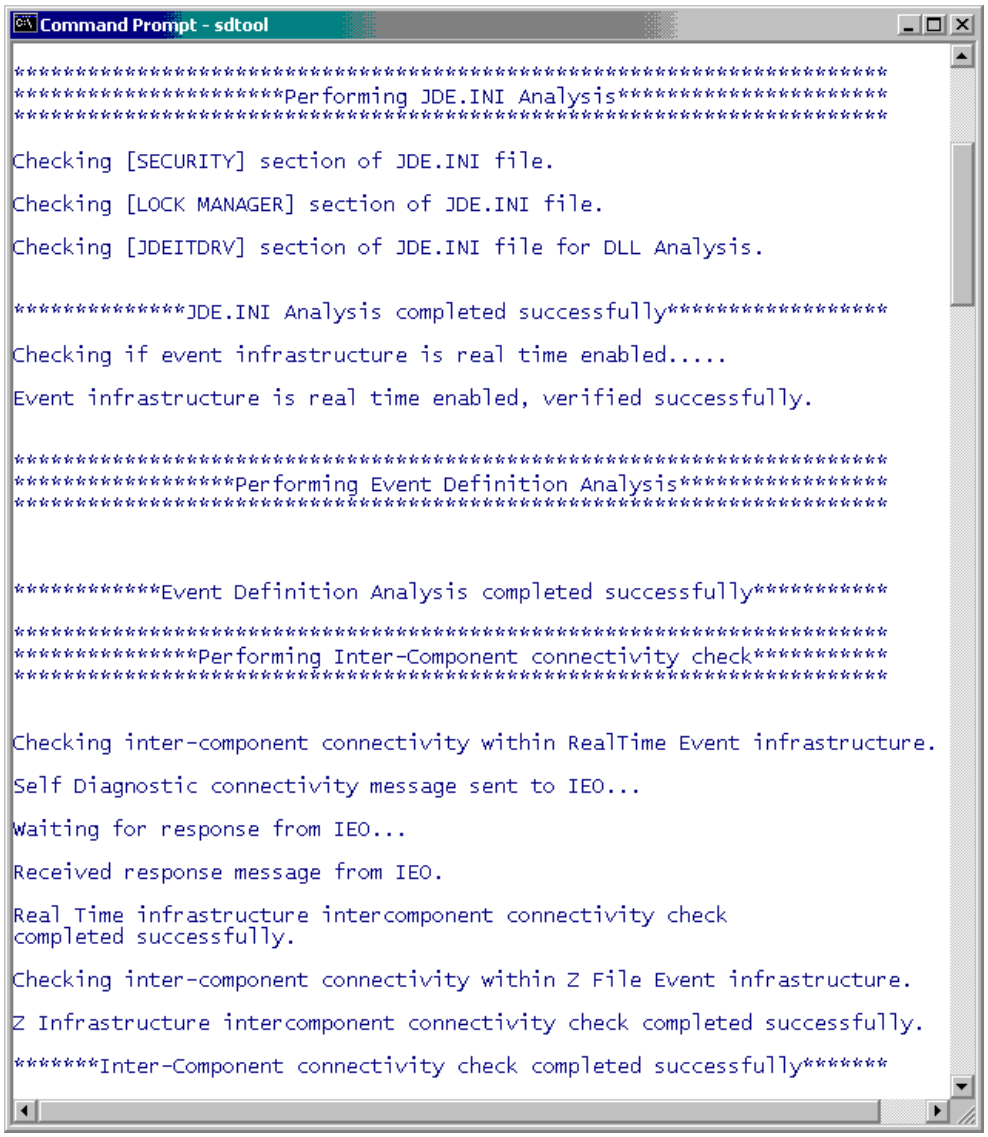
注:

$system は、使用するシステム上でアプリケーションをインストールした場所のパスを表します。

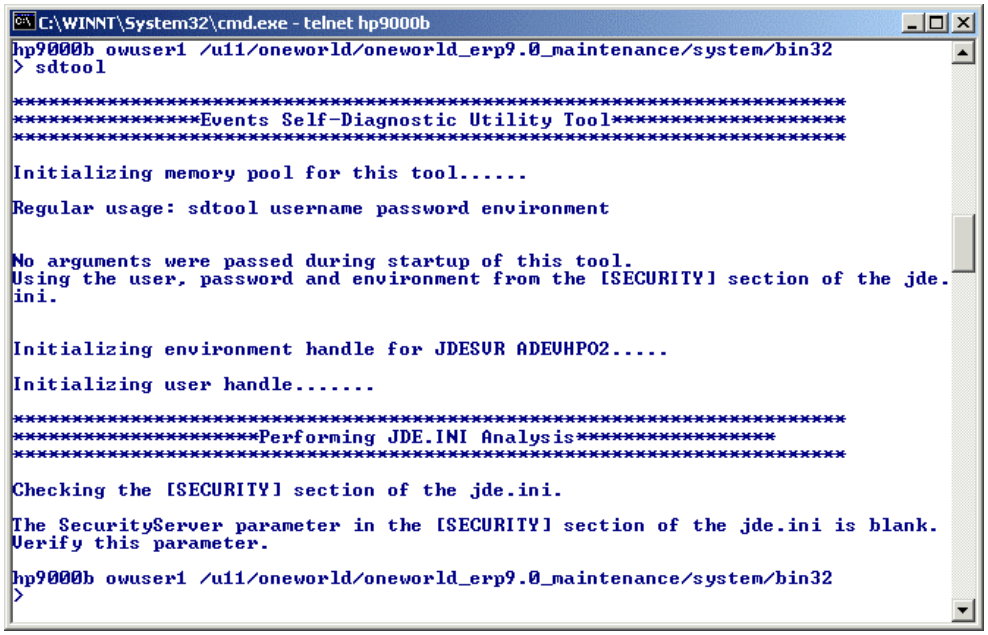
次の例に、ツールを ERP サーバーから起動する方法を示します。この例では、ツールは jde.ini ファイルの[SECURITY]セクションにアクセスして、有効なユーザー名、パスワード、および環境を読み取ります。



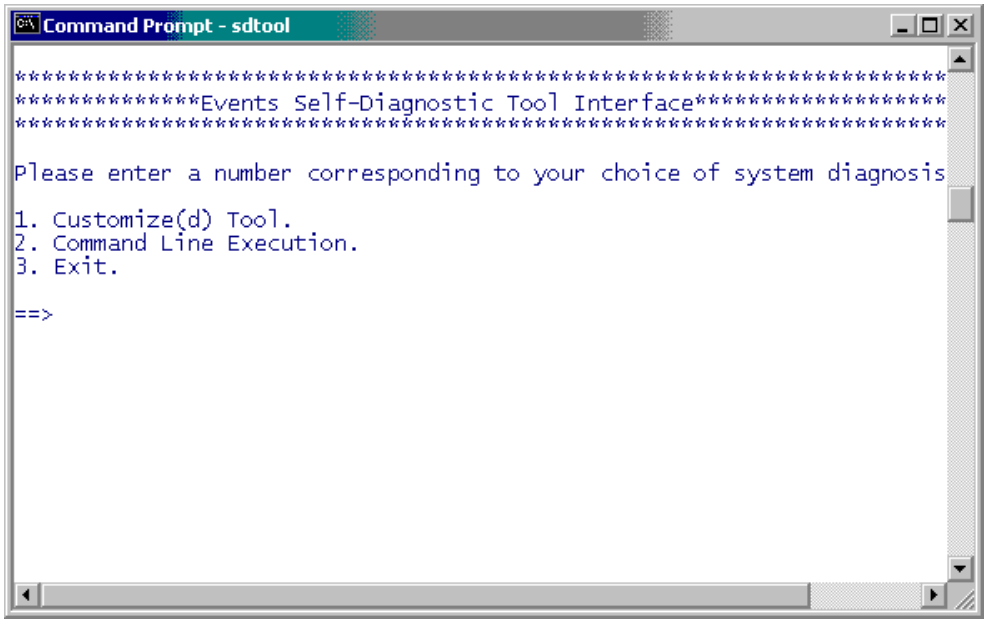
起動時に、イベント自己診断ツールは jde.ini ファイルを分析し、イベントが定義されているかどうかを検証します。また、イベント・インフラストラクチャ内部のコンポーネント間の接続性をチェックします。これらの各領域を分析するごとに、ツールはどの領域を分析中であるか、および分析が正しく行われたかどうかに関するフィードバックを提供します。次の例に、起動領域の診断が成功したというメッセージを示します。



起動領域のいずれかで問題を検出すると、ツールは診断を中止し、発生した問題を説明して問題の解決策を提案するメッセージを送ります。次の例に、ログオン情報が与えられておらず、jde.ini ファイル内で有効なユーザー名、パスワード、および環境が見つからなかった場合のエラー・メッセージを示します。

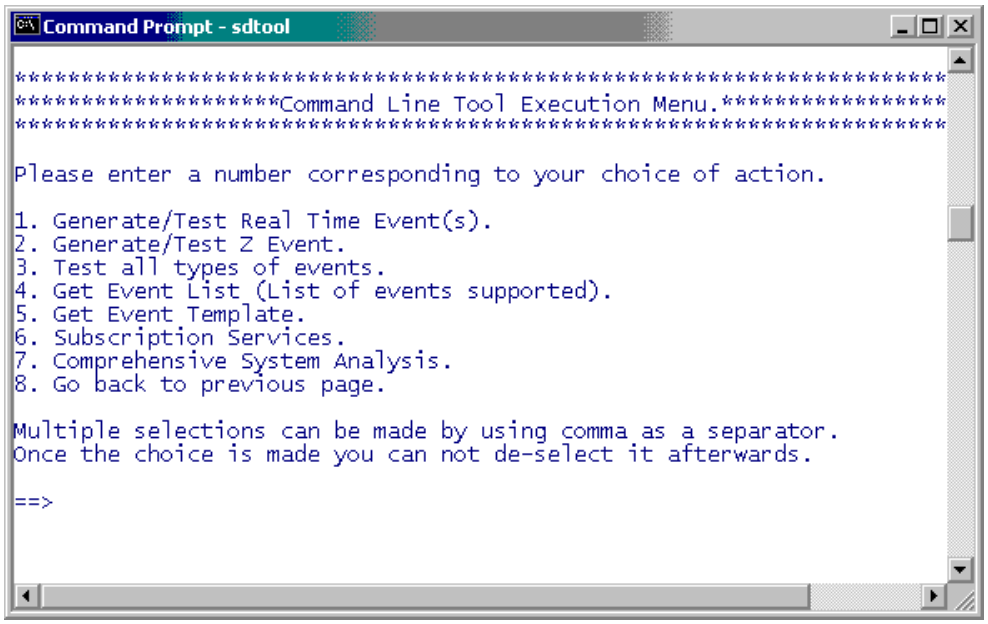


正常に起動した場合はオプションが表示され、カスタマイズされた構成ファイルを作成して使用するか、またはツールのコマンド行を使用して診断を実行するを選ぶことができます。[カスタマイズ済みツール]オプションを選ぶと、診断テストを作成してファイルに保存できます。この方法では、そのテストを必要に応じて実行でき、その都度ツールに情報を再入力する必要がなくなります。[コマンド行の実行]オプションを選んだ場合は、ツール・プロンプトに対してテスト情報を入力する必要があります。次の例に、テストの実行方法を指定するためのオプションを示します。コマンド行からテストを実行すると、結果ステートメントに続いて常にこのインターフェイス・メニューに表示されるため、別のテストを実行するかツールを終了できます。



オプション 1 の[カスタマイズ済みツール]とオプション 2 の[コマンド行の実行]のどちらを選んだ場合も、ツールでは同じテストが実行されます。次の例に、ツールでサポートされているテストを示しま

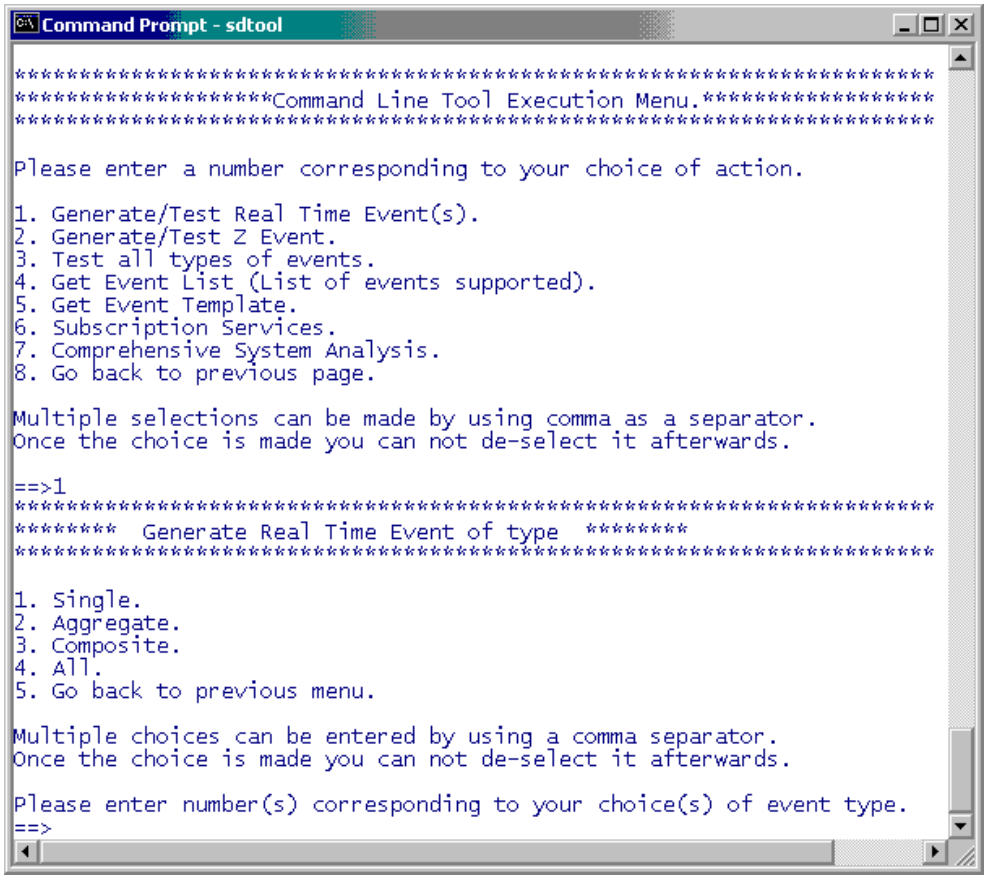
す。この例は[コマンド行の実行]オプションを選んだ場合を示します。ただし、[カスタマイズ済みツール]オプションを選んで新規ファイルを作成する場合も、処理メニューは同じです。



プロンプトに対してテスト番号を入力して 1 つ以上のテストを選び、[Enter]または[Return]キーを押します。複数のテストを選ぶ場合は、テスト番号をカンマ(,)で区切ってください。一部のテストには、さらにオプションが用意されています。プロンプトに対して、1 つまたは複数のオプションを入力します。複数のオプションを指定する場合はカンマで区切って入力してください。ツールでテストが実行され、成否を示すフィードバックが表示されます。テストに失敗した場合は、テスト失敗と詳細の検討が必要なログを示すフィードバックが表示されます。

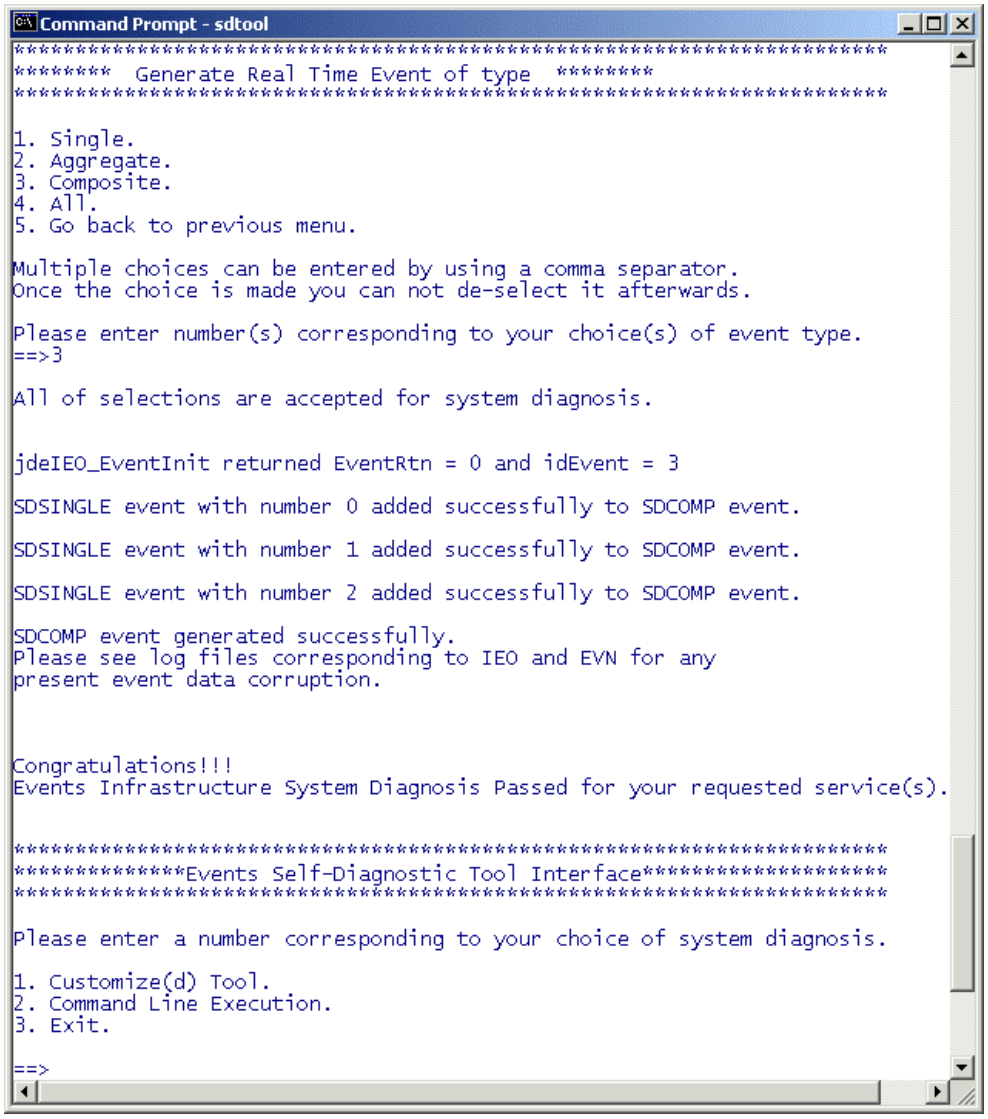
リアルタイム・イベントの生成/テスト

処理 1 の[Generate/Test Real Time Event(s)(リアルタイム・イベントの生成/テスト)]を選ぶと、リアルタイム・イベントのタイプが表示され、テスト対象として 1 つまたは複数のタイプを選ぶことができます。次の例に、リアルタイム・イベント・タイプのオプションを示します。



ツールでは、要求したリアルタイム・イベントが生成され、そのイベントに自己診断 XML ドキュメントが添付されます。イベント・インフラストラクチャおよびイベント・レシーバ・トランスポート・ドライバ内の各カーネルで、イベントの内容が添付されている XML ドキュメントと対照して検証され、データ破損がないかどうかがチェックされます。イベントが正常に成功されたかどうかを示すメッセージが表示されます。また、フィードバック・メッセージとして「Please see log files corresponding to IEO and EVN for any present event data corruption.(イベント・データの破損については IEO または EVN に対応するログ・ファイルを参照してください。)」も表示されます。このメッセージは、ログ・ファイルを調べて XML ドキュメントの不一致が発生したかどうかを判断するように指示するものです。最後に、指定した処理の分析が完了したことを示すメッセージが表示され、ツール・インターフェイス・メニューに戻ります。

次の例に、複合リアルタイム・イベントについて成功した診断を示します。

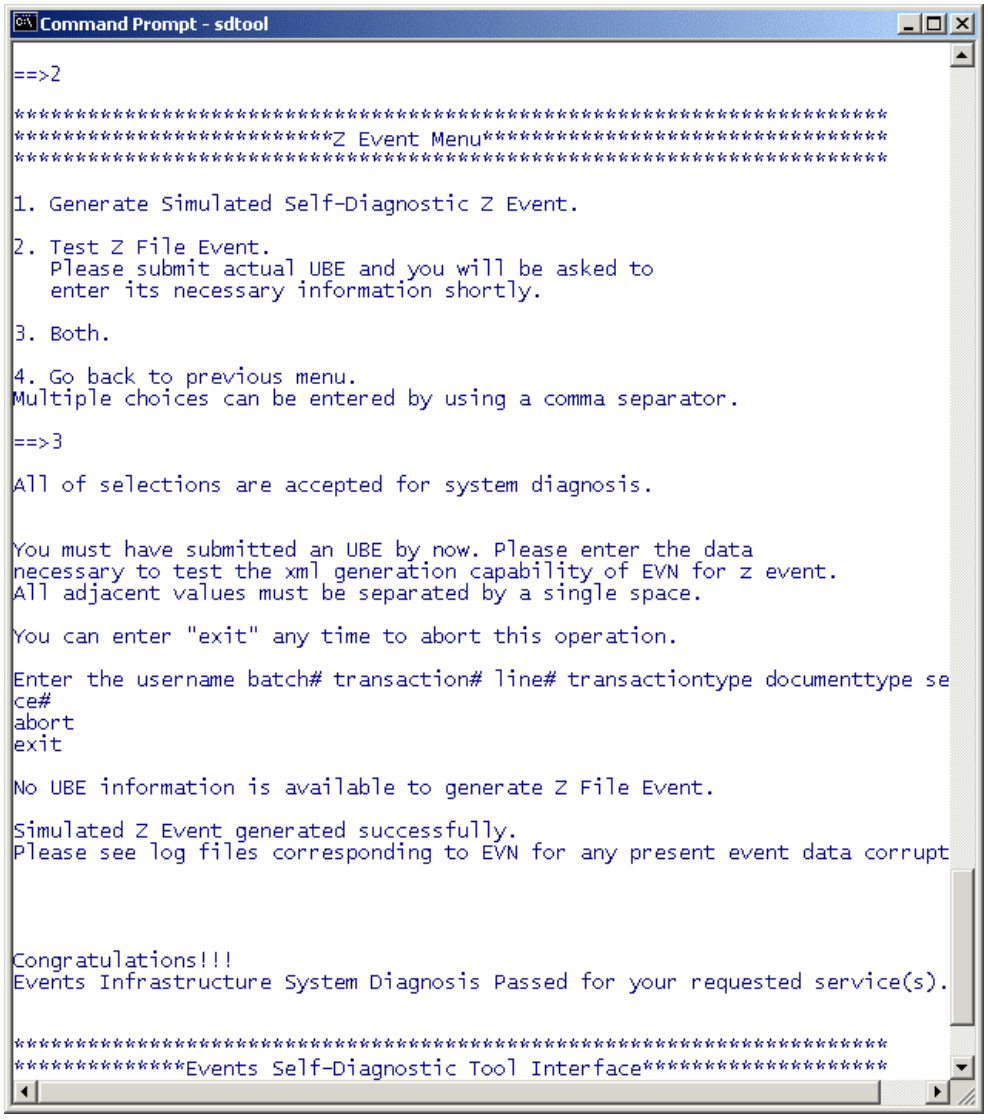


Z イベントの生成/テスト

処理 2 の[Generate/Test Z Event(Z イベントの生成/テスト)]を選ぶと、さらに一連のオプションが表示されます。シミュレートされた Z イベント、または実際のインターフェイス・テーブル(Z テーブル)を使用する Z イベントをテストできます。シミュレートされた Z イベントをテストするように選んだ場合は、そのイベントが生成され、自己診断 XML ドキュメントが添付されます。EVN カーネルとイベント・レシーバ・トランスポート・ドライバで、イベントの内容が添付されている XML ドキュメントと対照して検証され、データ破損がないかどうかがチェックされます。イベントが正常に成功されたかどうかを示すメッセージが表示されます。また、フィードバック・メッセージとして「Please see log files corresponding to EVN for any present data corruption.(データの破損については EVN に対応するログ・ファイルを参照してください。)」も表示されます。このメッセージは、EVN ログ・ファイルを調べて XML ドキュメントの不一致が発生したかどうかを判断するように指示するものです。最後に、指定した処理の分析が完了したことを示すメッセージが表示され、ツール・インターフェイス・メニューに戻ります。

実際の Z イベントを生成する場合は、有効な UBE を指定し、該当するインターフェイス・テーブルを設定する必要があります。ユーザー名、バッチ番号、トランザクション番号、行番号、トランザクション・タイプ、ドキュメント・タイプ、および順序番号の指定を求めるプロンプトが表示されます。このテストには、既存のインターフェイス・テーブル（Z テーブル）が使用されます。実際の Z イベントを要求すると、Z イベントは生成されますが、自己診断 XML ドキュメントは添付されません。EVN カーネルは、イベントの成否を示すメッセージを戻します。自己診断 XML ドキュメントがないため、ツールはデータ破損の有無を診断できません。

次の例に、シミュレートされた Z イベントと実際の Z イベントに対する診断要求を示します。実際の Z イベント情報を求めるプロンプトが表示された場合、ツールは Z ファイル生成/テストのこの部分で中止されています。シミュレートされた Z イベントのテストは完了しており、次のようなフィードバック情報が表示されます。



すべてのイベント・タイプのテスト

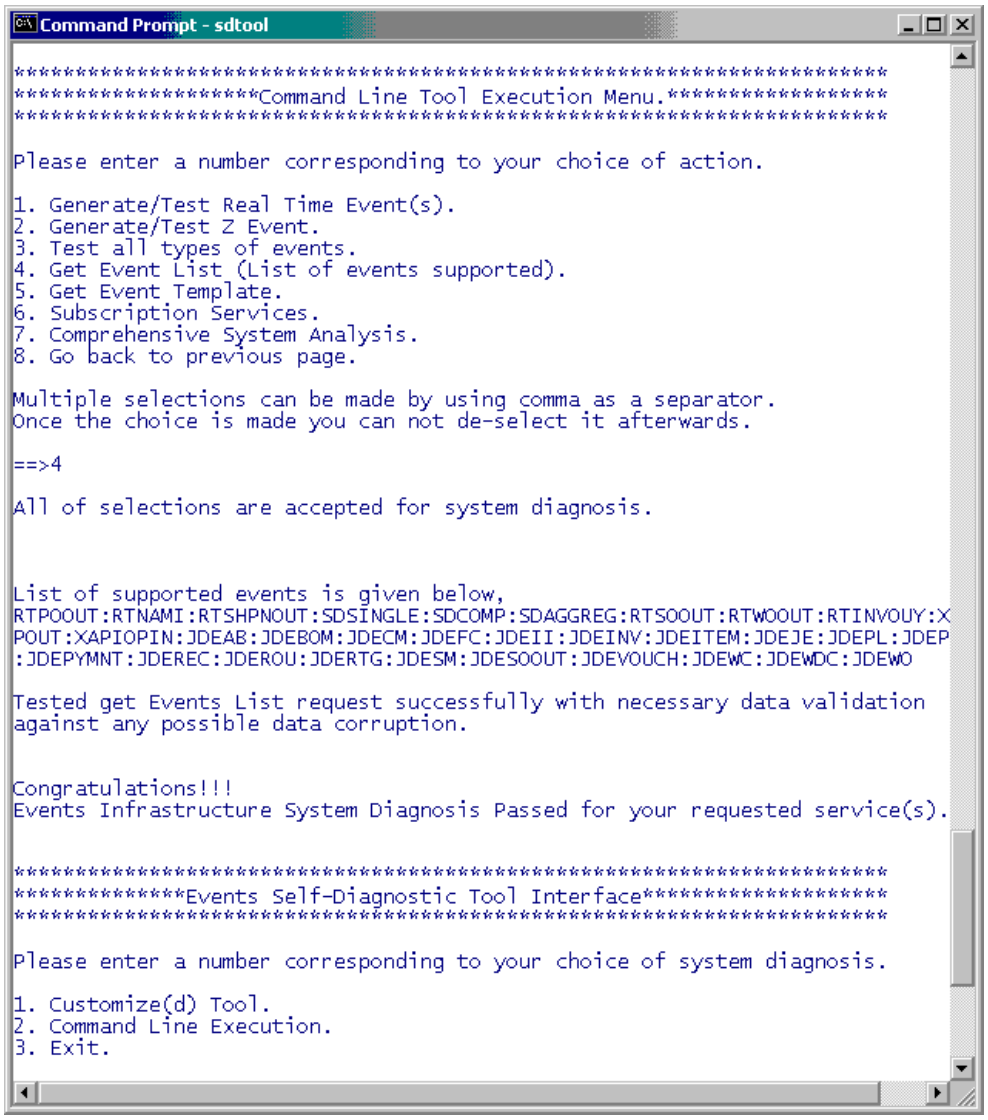
[Execution（実行）]メニューから処理 3 の[Test all types of events（すべてのイベント・タイプのテスト）]を選ぶと、すべてのリアルタイム・イベント（単一、集合体、複合）と両方の Z イベント（シミュレートと実際）がテストされます。処理 3 は、処理 1（3 つのオプションすべて）と処理 2（両方のオプション）の組合せに相当します。リアルタイム・イベントとシミュレートされた Z イベントについては、そのイベントが生成され、自己診断 XML ドキュメントが添付されるため、データの破損があれば検出できます。このオプションを選んだ場合は、有効な UBE を指定し、該当するインターフェイス・テーブルを設定する必要があります。このテストを実行しても実際の Z イベント・データがない場合は、Z イベント情報を要求された時点で"Exit"と入力して、テストのその部分を中止できます。

イベント情報がつるから IEO および EVN カーネルに送られ、それぞれのカーネルから各イベントの成否を示すメッセージが戻されます。

イベント・リストの取得

[Execution]メニューから処理 4 の[Get Event List (List of events supported)（イベントリストの取得 – サポートされていイベントのリスト）]を選ぶと、直ちにテストが実行されます。

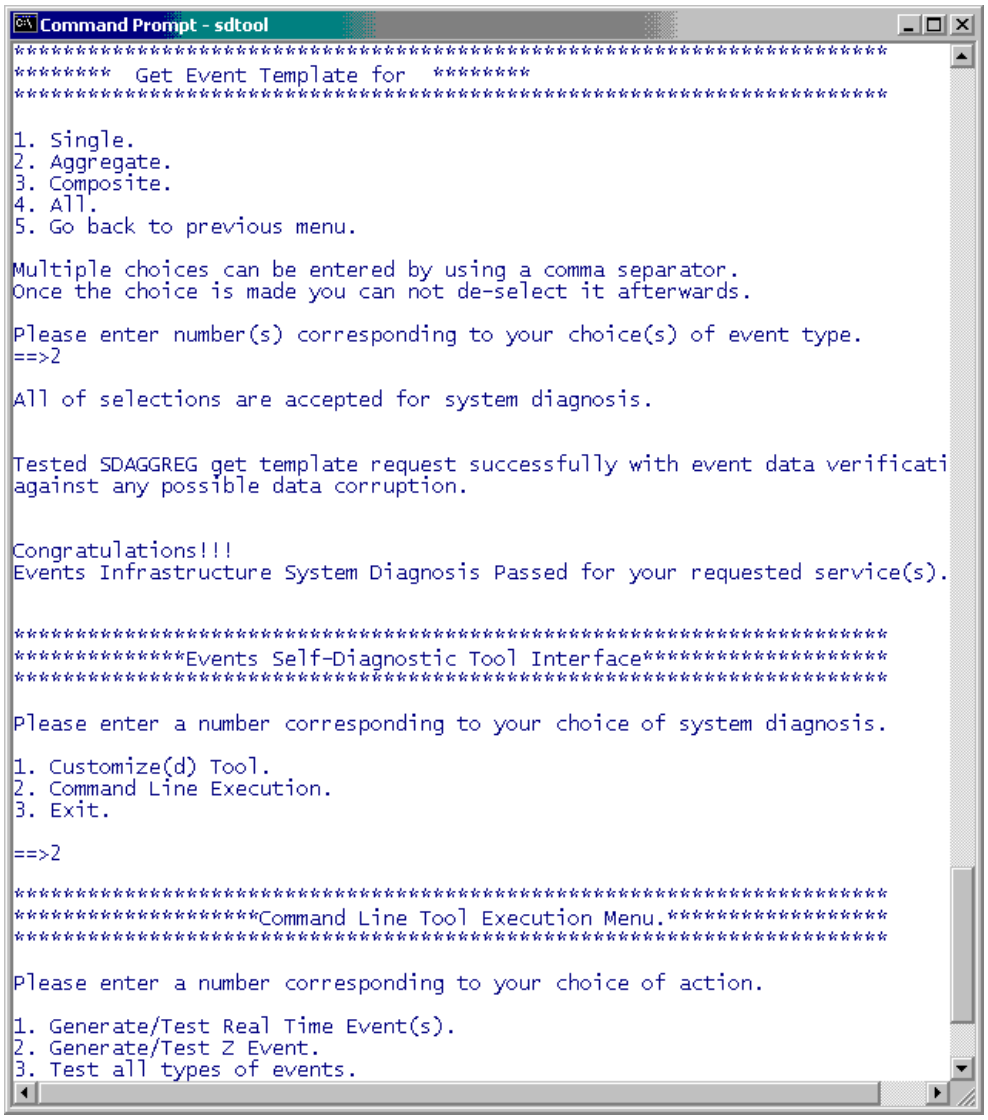
ツールから EVN カーネルに getEventList 要求が送られます。EVN カーネルは、定義されているイベント名のリストで応答します。リストが完全かどうかを確認するために、ツールではリストに自己診断イベント名があるかどうかがチェックされます。イベント・リストと共に、テストの成否を示すメッセージが表示されます。次のリストに、自己診断イベント SDSINGL、SDCOMP、SDAGGREG を含め、サポートされているイベントのリストを示します。



イベント・テンプレートの取得

処理 5 の[Get Event Template(イベント・テンプレートの取得)]を選ぶと、関連付けられているリアルタイム・イベントのタイプが表示されます。

ツールによりテンプレート要求が生成され、自己診断 XML ドキュメントが添付されます。イベント・インフラストラクチャおよびイベント・ジェネレータ内の各カーネルで、テンプレート要求が添付されている XML ドキュメントと対照して検証され、データ破損がないかどうかがチェックされます。その後、テンプレート要求に成功したことと、テンプレート・データが XML パケットと対照して確認されたことを示すフィードバックが表示されます。テストに失敗した場合は、その理由を示すメッセージが表示されます。次の例に、正常なテンプレート要求を示します。



サブスクリプション・サービス

処理 6 の[Subscription Services(サブスクリプション・サービス)]を選ぶと、一連のオプションが表示され、テスト対象となるサブスクリプション・サービスのタイプを選ぶことができます。

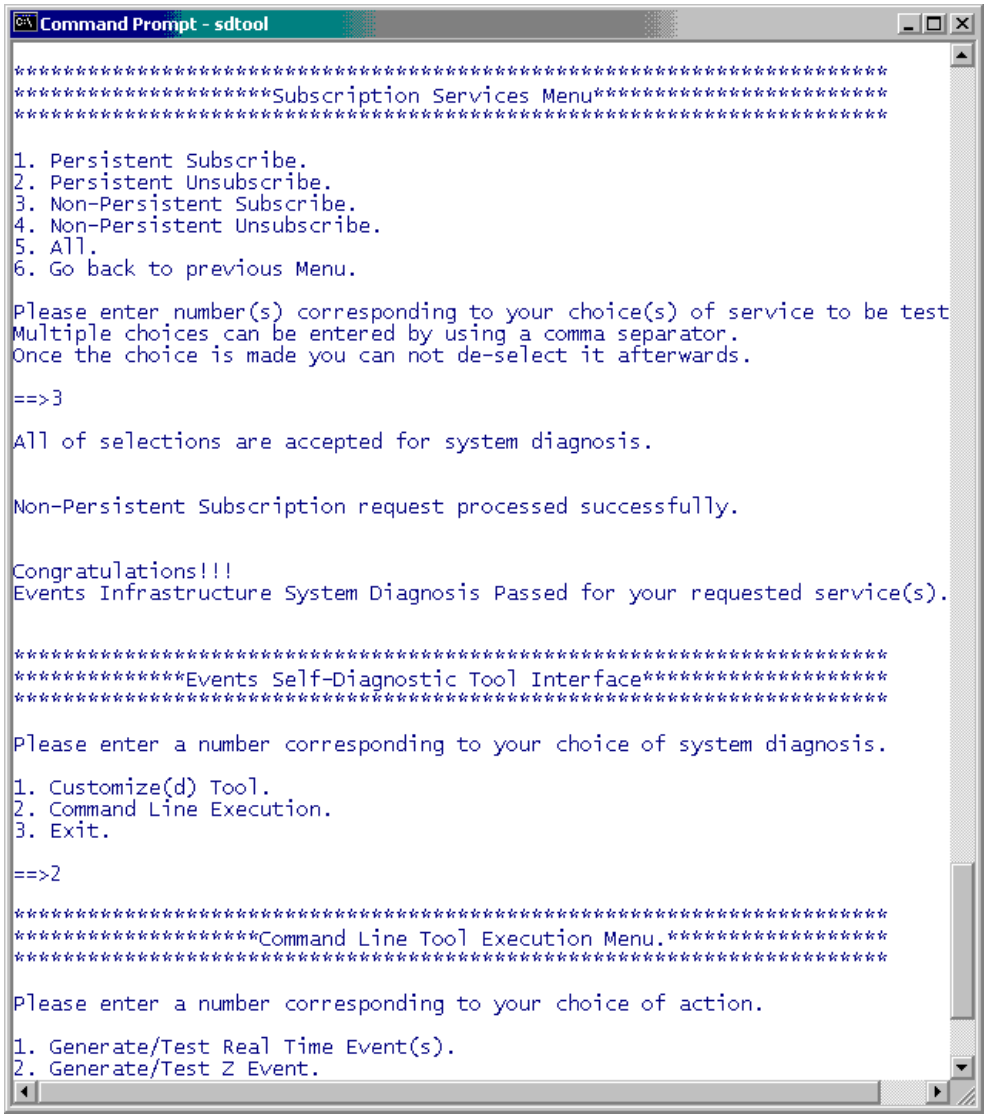
[Persistent Subscribe(持続的サブスクライブ)]オプションを選ぶと、登録されている自己診断イベントへの持続的サブスクリプション要求が EVN カーネルに送られます。サブスクライバ・リストは、システム構成に応じてファイルまたはデータベース・テーブル内で維持されています。このサブスクライバ・リストが確認され、テストの成否を示すメッセージが送信されます。

[Persistent Unsubscribe(持続的サブスクライブ解除)]オプションを選ぶと、登録されている自己診断イベントへの持続的サブスクリプション解除要求が EVN カーネルに送られます。システム構成に応じてファイルまたはデータベース・テーブル内にサブスクリプションがなくなったかどうかが確認され、テストの成否を示すメッセージが送信されます。

[Non-Persistent Subscribe(非持続的サブスクライブ)]オプションを選ぶと、登録されている自己診断イベントへの非持続的サブスクリプション要求が EVN カーネルに送られます。EVN カーネルにより維持されているサブスクライバ・リストが確認され、テストの成否を示すメッセージが送信されます。

[Non-Persistent Unsubscribe(非持続的サブスクライブ解除)]オプションを選ぶと、登録されている自己診断イベントへの非持続的サブスクリプション解除要求が EVN カーネルに送られます。サブスクリプションが EVN になくなったかどうかが確認され、テストの成否を示すメッセージが送信されます。

次の例に、[Non-Persistent Subscribe]オプションを選んだ場合の成功したテストを示します。



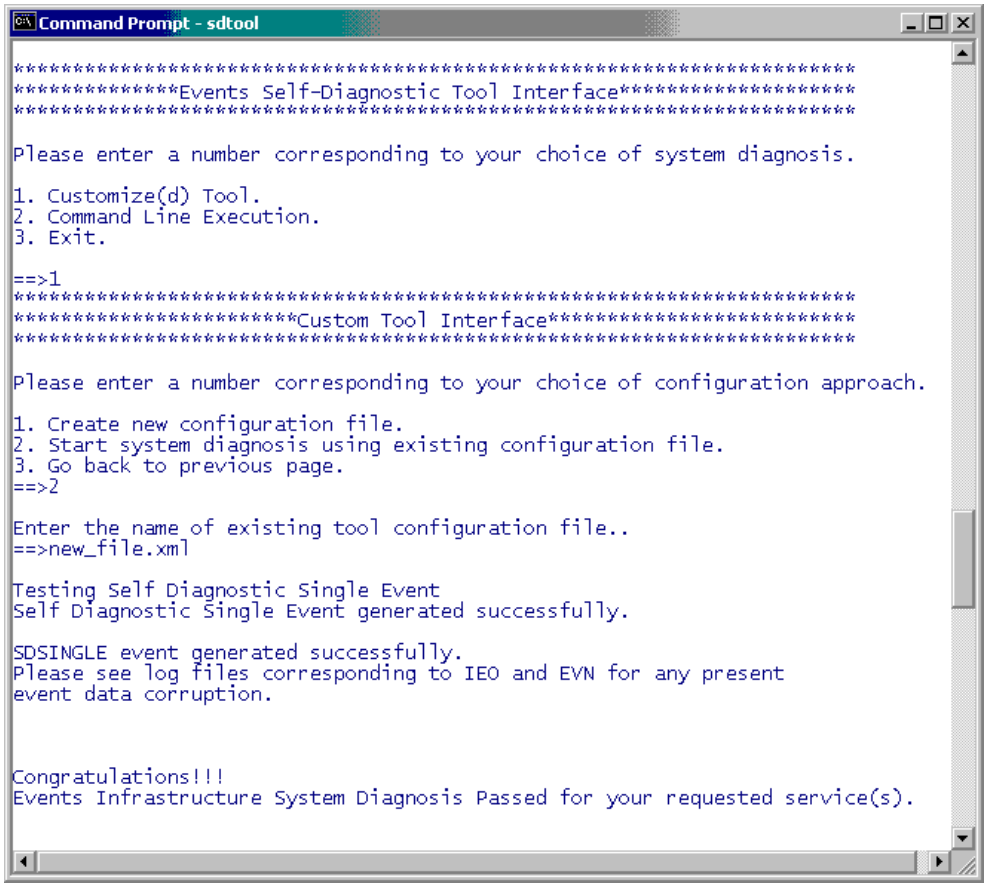
総合的なシステム分析

処理 7 の[Comprehensive System Analysis(総合的なシステム分析)]を選ぶと、すべてのテストが実行され、各テストの成否を示すメッセージが表示されます。

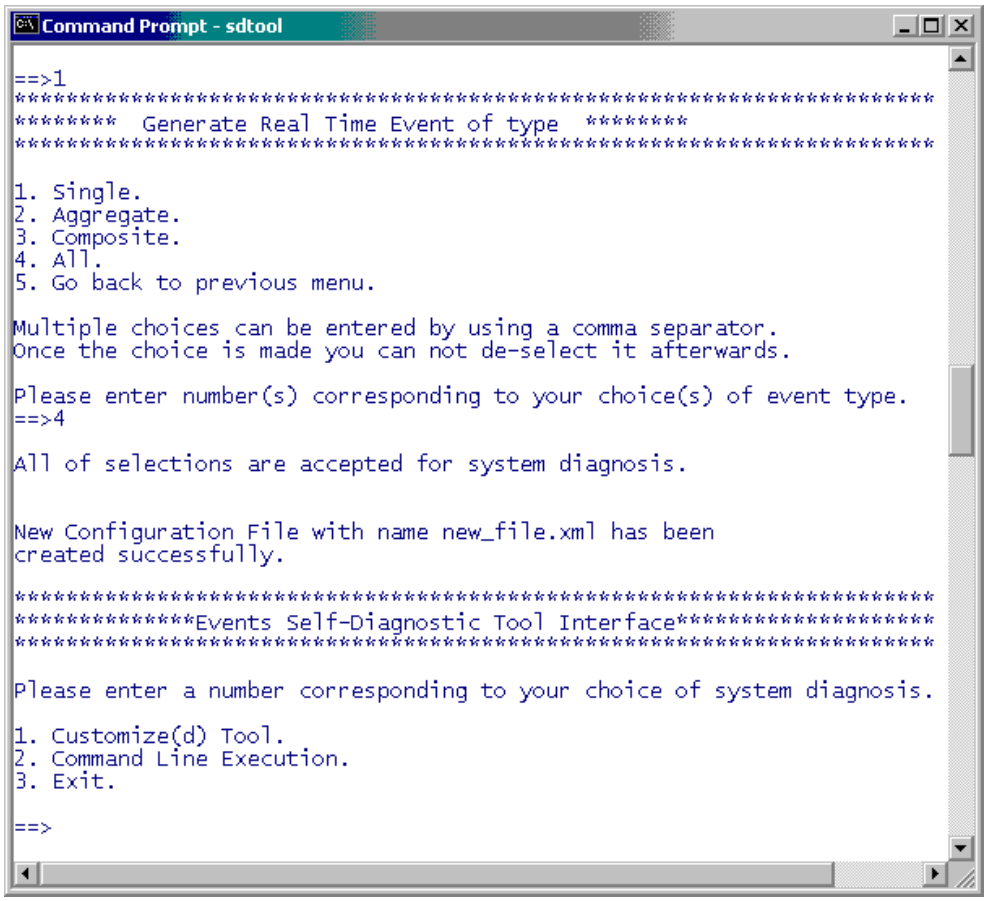
カスタマイズ済みツール

[Interface(インターフェイス)]メニューから[Customize(d) Tool]オプションを選ぶと、ファイル名を入力するか既存の構成ファイルを使用するかの選択を求めるプロンプトが表示されます。既存の構成ファイルは、このツールを使用して作成しておいたファイルです。テスト(処理)と各テストのオプションは、上記の場合と同じです。

既存の構成ファイル、つまり作成しておいた.xml ファイルを使用するには、プロンプトに対してファイル名を入力して[Enter]または[Return]キーを押します。以前に作成した設定ファイルに対する診断が開始されます。次の例に、単一リアルタイム・イベントの生成をテストする既存の構成ファイルの使用を示します。



新規の構成ファイルを作成するには、拡張子.xml を指定してファイル名を入力し、[Enter]または[Return]キーを押します。[Command Line Execution]メニューの場合と同じテストが用意されています。1 つまたは複数のテストを選ぶことができます。複数のテストを選ぶ場合は、テスト番号をカンマで区切って指定します。次の例に、単一、集合体、および複合リアルタイム・イベントの生成をテストする新規構成ファイルの作成方法を示します。


Command Prompt - sdtool
==>1
****
******** Generate Real Time Event of type ********
****
1. Single.
2. Aggregate.
3. Composite.
4. All.
5. Go back to previous menu.
Multiple choices can be entered by using a comma separator.
Once the choice is made you can not de-select it afterwards.
Please enter number(s) corresponding to your choice(s) of event type.
==>4
All of selections are accepted for system diagnosis.
New Configuration File with name new_file.xml has been
created successfully.
****
**************Events Self-Diagnostic Tool Interface*********************
****
Please enter a number corresponding to your choice of system diagnosis.
1. Customize(d) Tool.
2. Command Line Execution.
3. Exit.
==>

バッチ・インターフェイス

通常は、バッチ・インターフェイスを使用して、期間中のトランザクションを収集してから、すべてのトランザクションを一度に処理します。J.D. Edwards では、次のバッチ・インターフェイス・モデル・タイプをサポートしています。

- インターフェイス・テーブル
- 電子データ・インターフェイス（EDI）
- テーブル変換
- 出力ストリーム・アクセス(OSA) UBE
- 拡張プランニング・エージェント(APAg)/インテグレーション

インターフェイス・テーブル

インターフェイス・テーブルは、ERP 以外の情報を保管して ERP に対して処理できるワークテーブルであり、Z テーブルとも呼ばれます。インターフェイス・テーブルを使用して ERP データを取り込むこともできます。J.D. Edwards インターフェイス・テーブルにより、ERP アプリケーション・テーブルがミラー化されます。インターフェイス・テーブルでは、次の機能を使用できます。

- ビジネス関数に関する規則
- Z トランザクション
- フラット・ファイル

一部のアプリケーションの場合は、トランザクション用に事前定義済みのインターフェイス・テーブルが用意されています。J.D. Edwards 標準フォーマットを使用すれば、独自のインターフェイス・テーブルを作成することもできます。

参照

- 事前定義済みのインターフェイス・テーブルとプロセスのリストについては、『インタオペラビリティ』ガイドの「インターフェイス・テーブル」

インターフェイス・テーブルの構造

各トランザクションには、一連のインターフェイス・テーブルが使用されます。インターフェイス・テーブルの共通セットを使用するファイルもあります。インターフェイス・テーブル名は、ERP アプリケーションのテーブル名に基づき、サフィックス Z1 が付いています。たとえば、アプリケーション・テーブルが F4211 の場合、インターフェイス・テーブルは F4211Z1 となります。

次のガイドラインに従って、基準となるテーブルを確定してください。

- 受信は、インターフェイス・テーブルからのデータで更新されるアプリケーション・テーブルに基づいています。
- 送信は、インターフェイス・テーブルに入れるデータの抽出元となるアプリケーション・テーブルに基づいています。

システムの内部トランザクションの場合は、送受信の両方向に 1 組のインターフェイス・テーブルが使用されます。たとえば、受注オーダー用の受注オーダー・システムでは、この標準に従って受信得意先オーダー(850)および送信オーダー承認(855)が 1 組のインターフェイス・テーブルを共有します。

インターフェイス・テーブルが受信トランザクションおよび送信トランザクション両方に使用される場合は、基準となるテーブルが同じアプリケーション・テーブルであることが必要です。受信得意先オーダーおよび送信オーダー承認がある上述の受注オーダーの例では、詳細インターフェイス・テーブルは、受注明細テーブル(F4211)に基づいています。

インターフェイス・テーブルのカラム数が 250 を超え、レコードの長さが 1968 より大きい場合、残りのカラム用に追加のインターフェイス・テーブルが必要になります。追加のインターフェイス・テーブルのカラムには、まれに使用されるデータが含まれるはずです。この追加のインターフェイス・テーブルは、サフィックス Z1 の後の文字が A で始まるプライマリ・インターフェイス・テーブルにちなんで命名されます。たとえば、プライマリ・インターフェイス・テーブルが F4211Z1 の場合、追加のテーブルは F4211Z1A になります。

テーブルの先頭は次のカラムになり、制御フィールドとして機能します。

- User ID(ユーザー ID)(EDUS)- キーフィールド
- Batch Number(バッチ番号)(EDBT)- キーフィールド
- Transaction Number(トランザクション番号)(EDTN)- キーフィールド
- Line Number(行番号)(EDLN)- キーフィールド
- Document Type(伝票タイプ)(EDCT)
- Transaction Type(トランザクション・タイプ)(TYTN)
- Translation Format(変換フォーマット)(EDFT)
- Transmission Date(伝送日付)(EDDT)
- Direction Indicator(送受信インジケータ)(DRIN)
- Number of Detail Lines(明細行の番号)(EDDL)
- Processed(処理済み)(EDSP)
- Trading Partner ID(取引先 ID)(PNID)
- Action Code(アクションコード)(TNAC)

上記のキー構造体を使用する必要があります。

テーブルの最後には、ユーザー・フィールドおよび監査フィールドに予約された次の各カラムがあります。

- User Reserved Code(ユーザー用確保コード)(URCD)
- User Reserved Date(ユーザー用確保日付)(URDT)
- User Reserved Amount(ユーザー用確保金額)(URAT)
- User Reserved Number(ユーザー用確保番号)(URAB)
- User Reserved Reference(ユーザー用確保参照)(URRF)
- Transaction Originator(トランザクション入力者)(TORG)
- User ID(ユーザー ID)(USER)

- Program ID（プログラム ID）(PID)
- Work Station ID（ワークステーション ID）(JOBN)
- Date Updated（更新日付）(UPMJ)
- Time of Day（時間）(TDAY)

テーブルの中央には、ユーザー用確保カラムおよび監査フィールド・カラムを除き、基準となるアプリケーション・テーブルからのすべてのカラムがあります。これの例外は、インターフェイス・テーブルが 250 カラムの限界と 1968 レコードの長さの限界に近い場合です。この場合、ほとんど必要ないと思われるアプリケーション・テーブルからのカラムは除外されるはずです。

テーブル・カラムのプレフィックスは、見出しの場合は SY、明細の場合は SZ になります。

現金入金確認または入荷確認などの変更または照合インターフェイス・テーブルには、対話型フォームでユーザー入力制御に対応する追加カラムが必要になることがあります。

見出しテーブルは、トランザクションに必ずしも必要ではありません。

注:

カスタム・インターフェイス・テーブルを作成する場合は、上記の構造とフォーマットを使用してください。

受信インターフェイス・テーブルの処理

インターフェイス・テーブルを使用して、ERP 以外のトランザクションを既存の ERP データベースにインポートします。受信インターフェイス・テーブルは、トランザクションが保管される ERP アプリケーション・テーブルに基づいています。ERP では、受信インターフェイス・テーブルからのレコードを次の方法で処理できます。

- 受信バッチ処理の実行
- ビジネス関数の使用
- サブシステムを使用したトランザクションのトリガー

注:

ERP 以外のトランザクションは Z トランザクションと呼ばれます。ERP 以外の情報を ERP に対して処理する方法については、『インタオペラビリティ』ガイドの「Z トランザクション」を参照してください。

参照

- ERP 以外のトランザクションを ERP に転送する方法については、『インタオペラビリティ』ガイドの「Z トランザクション」

送信インターフェイス・テーブルの処理

インターフェイス・テーブルを使用して ERP から情報を取り込むことができます。送信インターフェイス・テーブルは、データの抽出元となる ERP アプリケーション・テーブルに基づいています。次のいずれかの方法で ERP からレコードを取り込むことができます。

- 抽出バッチ処理の実行
- ベンダ固有の送信バッチ処理の使用
- サブシステム・ビジネス関数の使用

参照

- 一度に 1 つずつトランザクションを処理するように送信マスター・ビジネス関数の処理オプションを設定して、ERP から情報を取り込む方法については、『インタオペラビリティ』ガイドの「Z イベント」

抽出バッチ処理の実行

ERP からのバッチ・トランザクションを使用すると、大量のトランザクションをインターフェイス・テーブルに入れて一括処理できます。送信するドキュメントのタイプ専用に設定された抽出バッチ処理を使用して、J.D. Edwards アプリケーション・テーブルから J.D. Edwards 送信インターフェイス・テーブルにレコードをコピーします。

抽出バッチ処理をサポートしているアプリケーションの場合は、Solution Explorer を使用して抽出バッチ処理を開始します。抽出バッチ処理は、レポート機能のバージョン・リストを表示します。既存のバージョンをそのまま実行したり、既存のバージョンを変更したり、バージョンを追加できます。バージョンのデータ選択および処理オプションを希望に合わせて変更することもできます。

抽出バッチ処理を実行すると、J.D. Edwards アプリケーション・テーブルからトランザクションのデータが取り込まれ、そのデータがインターフェイス・テーブルにコピーされます。また、処理された伝票をリスト表示する監査レポートも生成されます。エラーは監査レポートに出力され、〈従業員ワークセンター〉にも送信されます。改訂アプリケーションを使用すると、インターフェイス・テーブルのレコードを追加、変更、または削除できます。

▶ 抽出バッチ処理を実行するには

エクスポートしたいトランザクション・タイプについて、抽出バッチ処理を開始します。たとえば、〈作業オーダーの作業指示〉プログラムには抽出バッチ処理があります。抽出バッチ処理が用意されているプログラムについては、この項の末尾の「インタオペラビリティ・インターフェイス・テーブル情報」を参照してください。

1. 〈バージョン・プロンプト〉で、次のいずれかをクリックし、レポート機能オプションを検討します。
   - データ選択
   - データ順序
2. [投入]ボタンをクリックします。

抽出バッチ処理では、J.D. Edwards アプリケーション・テーブルからデータが取り込まれ、そのデータが送信インターフェイス・テーブルにコピーされます。

〈抽出バッチ処理〉プログラムは、処理が正常終了した伝票をリスト表示する監査レポートを生成します。

ベンダ固有の送信バッチ処理

ベンダ固有の送信バッチ処理の目的は、インターフェイス・テーブルのレコードをバッチ・モードで処理することです。バッチ処理は〈インタオペラビリティ汎用送信サブシステム UBE〉レポート(R00460)によって呼び出され、インターフェイス・テーブルのレコードに対するキーを受け取ります。

各ベンダ固有のバッチ処理は、処理されるトランザクションに特有のものです。データベース・レコード・データで実際に実行するものを判別する必要があります。バッチ処理は、J.D. Edwards ツール・セットを使用してバッチ処理がスペックに書き込まれます。ただし、J.D. Edwards により定義された次のデータ構造体を使用する必要があります。

データ項目	必須	I/O	説明
EDUS	Y	I	ユーザーID
EDBT	Y	I	バッチ番号
EDTN	Y	I	トランザクション番号
FFEM	N	I	フラット・ファイル・エクスポートモード
EDEM	N	I	外部データベース・エクスポート・モード
EAEM	N	I	外部 API エクスポート・モード
ERRC	N	O	エラー・コード
EDLN	N	I	行番号

サブシステム・ビジネス関数

汎用送信サブシステム・ビジネス関数 Add Transaction To Subsystem Queue(B0000176)を使用すると、レコードをサブシステム・データ待ち行列に書き込んで、サブシステム内で認識する必要のあるバッチ処理を指定できます。このビジネス関数は、バッチ・オブ・ワン(単一トランザクション)の処理を開始します。また、ビジネス関数はキー・データをサブシステム・データ待ち行列に渡します。

送信トランザクションのデータ構造体は次のとおりです。

行番号	EDLN
トランザクション・タイプ	TYTN
伝票タイプ	DCTO
アクション・コード	TNAC

インターフェイス・テーブルのメンテナンス

インターフェイス・テーブルの処理中にエラー・メッセージが表示された場合は、改訂アプリケーションを使用してデータを修正してから、そのデータをバッチ・モードまたはトランザクション・モードで処理できます。インターフェイス・テーブル内のデータが正常に処理された後、除去プログラムを実行してインターフェイス・テーブルからレコードをすべて削除し、システムからセカンダリ・インターフェイス・テーブルを削除する必要があります。

改訂アプリケーション

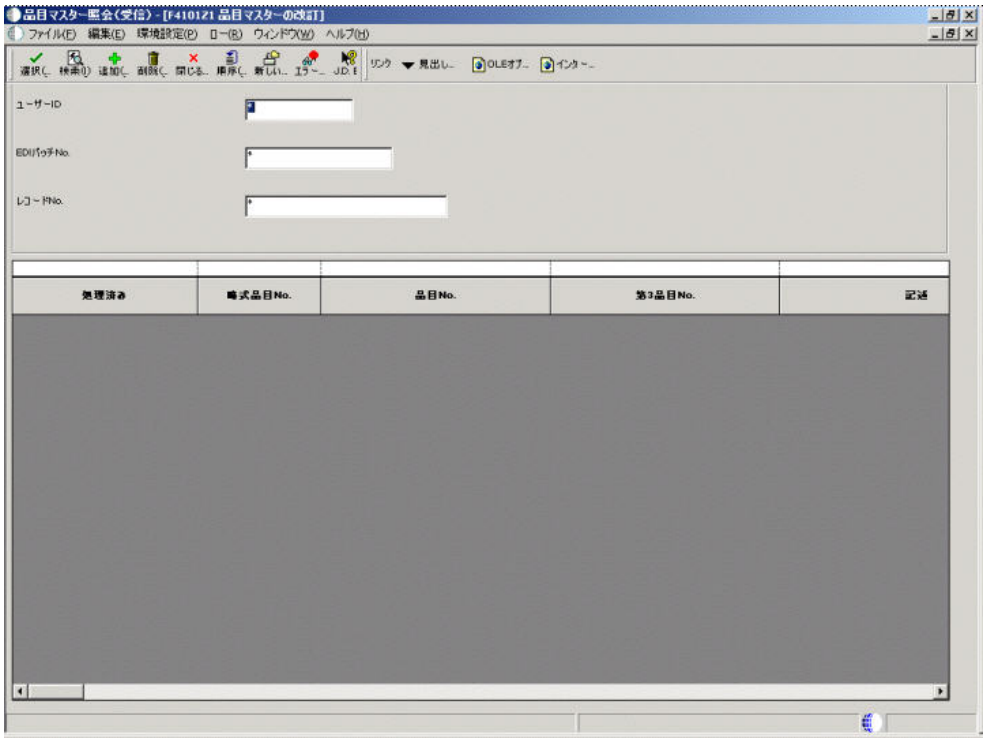
インターフェイス・テーブルのトランザクションを追加、削除、編集および検討するには、改訂アプリケーションを使用します。処理しているトランザクションのタイプに該当する改訂アプリケーションを使用してください。改訂アプリケーションを使用すると、エラーのあるレコードを修正できます。その後、処理をもう一度実行し、引続き修正を行い、エラーが発生することなくプログラムが終了するまでトランザクション処理を再実行します。レコードを削除する場合、改訂アプリケーションが該当する除去イベント・ルール・ビジネス関数を呼び出してレコードを削除します。名前は、明細インターフェイス・テーブルに基づいています。たとえば、受注入力のテーブルが F4201Z1 および F4211Z1 であった場合、アプリケーションは P4211Z1 となります。

参照

- J.D. Edwards の事前定義済みのインターフェイス・テーブルと共に使用できる改訂アプリケーションのリストについては、『インタオペラビリティ』ガイドの「インタオペラビリティ・インターフェイス・テーブル情報」

▶ 受信トランザクションを検討し改訂するには

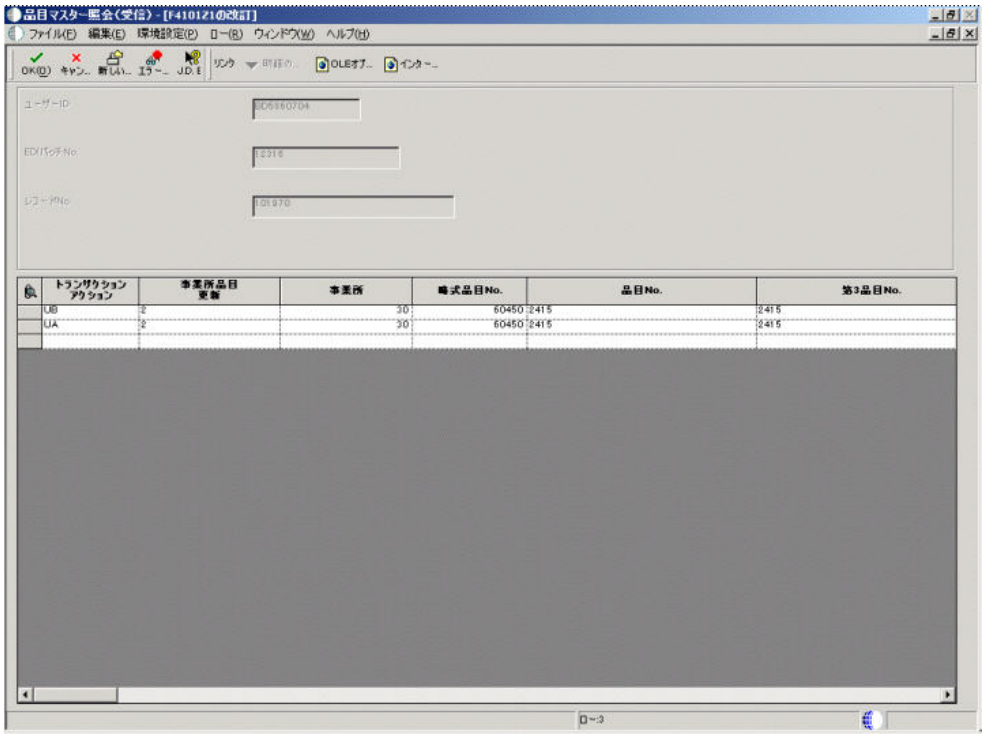
〈品目マスター改訂の処理〉など、インタオペラビリティをサポートしているアプリケーションから、使用する改訂アプリケーションを選びます。



1. 使用する改訂アプリケーション(品目マスター改訂の処理など)で、検索を特定のトランザクションに制限するには、次のフィールドに値を入力します。
   - ユーザーID
   - バッチ番号

- トランザクション番号

2. [検索]をクリックします。
3. トランザクションを選択し、[選択]をクリックします。



4. 〈改訂〉で、トランザクションを検討、改訂し、[OK]をクリックします。
5. 適用可能な場合は、[ロー]メニューから[明細の改訂]を選択し、追加詳細データを検討または変更し、終了した時点で[OK]をクリックします。

受信トランザクション処理が識別したエラーを修正した後、もう一度トランザクション処理を実行します。その他のエラーが識別される場合は、エラーを訂正し、もう一度トランザクション処理を実行します。

フィールド記述

記述	用語解説
ユーザーID	取引(トランザクション)データの作成元を示します。これはユーザーID、端末 ID、外部システム・アドレス、ネットワーク・ノードなどです。このフィールドは入力データと送信元の両方の識別に役立ちます。
EDI バッチ No.	システムがバッチに割り当てる番号。バッチ処理でユーザーが作成した各バッチに自動的に採番されます。
レコード No.	EDI(電子データ交換)システムでトランザクションに割り当てられる番号。EDI を使用していない環境では、固有の ID 番号を割り当ててください。伝票番号と同じ番号を使用することもできます。

インターフェイス・テーブル情報の除去

インターフェイス・テーブル内のデータを正常に処理した後で、除去バッチ処理を定期的に実行する必要があります。除去バッチ処理には、1 つまたは複数のセクションが必要です。このセクション数は、インターフェイス・テーブルに応じて異なります。除去用バッチ処理は除去用 NER を呼び出します。除去用バッチ処理の名前は、P のサフィックスを持つ改訂アプリケーションに基づいています。たとえば、改訂アプリケーションが P4211Z1 の場合、除去用バッチ処理は R4211Z1P になります。

除去用イベント・ルール・ビジネス関数には2つのモードがあります。

- 見出しモード。見出しレコードおよび依存しないテーブルにある関連レコードすべてを削除します。
- 明細モード。詳細レコードおよび依存するテーブルにある関連レコードすべてを削除します。

除去用 NER は除去用バッチ処理にちなんで命名されます。NER 名には 8 文字しか許可されていません。標準を使用して名前に 9 文字を使用した場合は、P のサフィックスを削除します。たとえば、除去用バッチ処理が R4211Z1P の場合、除去 NER は N4211Z1P になります。

正味変更に対する変更前トランザクションが削除される場合、該当する変更後トランザクションも削除されます。変更後トランザクションが削除される場合、該当する変更前トランザクションも削除されます。

参照

- 除去バッチ処理のリストについては『インタオペラビリティ』ガイドの「インタオペラビリティ・インターフェイス・テーブル情報」

インタオペラビリティ・インターフェイス・テーブル情報

アプリケーション	インターフェイス・テーブル (Z テーブル)	入力サブシステム・バッチ・プロセス	入力プロセッサ・バッチ・プロセス	抽出バッチ・プロセス	改訂アプリケーション	除去バッチ・プロセス	アプリケーション (処理オプションあり)
会計							
住所録	F0101Z2	R01010Z – ZJDE0002	R01010Z – ZJDE0001		P0101Z1	R0101Z1P	P0100041
顧客マスター	F03012Z1	R03010Z – ZJDE0002	R03010Z – ZJDE0001		P0301Z1	R0101Z1P	P0100042
仕入先マスター	F0401Z1	R04010Z – ZJDE0002	R04010Z – ZJDE0001		P0401Z1	R0101Z1P	P0100043
売掛請求書	F03B11Z1, F0911Z1, F0911Z1T	R03B11Z1I	R03B11Z1I – ZJDE0001	該当なし	P03B11Z1	R03B11Z1P	該当なし
買掛請求書	F0411Z1, F0911Z1	R04110Z – ZJDE0002	R04110Z – ZJDE0001	該当なし	P0411Z1	R0411Z1P	該当なし
支払オーダー (送金あり)	F0413Z1, F0414Z1	該当なし	該当なし		P0413Z1	R0413Z1	P0413M

アプリケーション	インターフェイス・テーブル (Zテーブル)	入力サブシステム・バッチ・プロセス	入力プロセッサ・バッチ・プロセス	抽出バッチ・プロセス	改訂アプリケーション	除去バッチ・プロセス	アプリケーション (処理オプションあり)
仕訳	F0911Z1, F0911Z1T	R09110Z – ZJDE0005	R09110Z – ZJDE0002		P0911Z1	R0911Z1P	該当なし
固定資産マスター	F1201Z1, F1217Z1	R1201Z1I – XJDE0002	R1201Z1I – XJDE0001	R1201Z1X	P1201Z1	R1201Z1P	P1201
勘定残高	F0902Z1	該当なし		該当なし	P0902Z1	R0902ZP	該当なし
バッチ入金(現金)	F03B13Z1	該当なし	R03B13Z1I – ZJDE0001	該当なし			該当なし
HRM							
給与計算時間入力	F06116Z1	R05116Z1I	R05116Z1I – ZJDE0001	該当なし	P05116Z1	R05116Z1P	該当なし
流通							
購買オーダー	F4301Z1, F4311Z1	R4311Z1I – XJDE0002	R4311Z1I – XJDE0001		P4311Z1	R4301Z1P	P4310
送信入荷確認	F43121Z1	該当なし	該当なし		P43121Z1	R43121Z1P	P4312
入荷工程	F43092Z1	R43092Z1I – XJDE0002	R43092Z1I – ZJDE0001		P43092Z1	R43092Z1P	P43250
送信受注オーダー	F4201Z1, F4211Z1, F49211Z1	該当なし	該当なし		P4211Z1	R4211Z1P	P4210
送信出荷確認	F4201Z1, F4211Z1, F49211Z1	該当なし	該当なし		P4211Z1	R4211Z1P	P4205
ロジスティクス							
循環棚卸	F4141Z1	R4141Z1I	R4141Z1I – ZJDE0001	該当なし	P4141Z1	R4141Z1P	該当なし
品目マスター	F4101Z1, F4101Z1A	R4101Z1I	R4101Z1I – ZJDE0001		P4101Z1		P4101
品目原価	F4105Z1	該当なし	R4105Z1I – XJDE0001		P4105Z1	R4105Z1P	P4105
倉庫確認(提示)	F4611Z1	R4611Z1I	R4611Z1I – ZJDE0001		P4611Z1	R4611Z1P	該当なし

アプリケーション	インターフェイス・テーブル (Zテーブル)	入力サブシステム・バッチ・プロセス	入力プロセッサ・バッチ・プロセス	抽出バッチ・プロセス	改訂アプリケーション	除去バッチ・プロセス	アプリケーション (処理オプションあり)
製造							
作業オーダー見出し	F4801Z1	作業オーダー完了を使用	作業オーダー完了を使用	R4101Z1O	P4801Z1	R4801Z1P	P48013
作業オーダー部品リスト	F3111Z1	計画メッセージを使用	計画メッセージを使用		P4801Z1	R3111Z1P	P3111
作業オーダー作業工程	F3112Z1	計画メッセージを使用	計画メッセージを使用	R4801Z2X	P4801Z1	R3112Z1P	P3112
作業オーダー従業員時間入力	F31122Z1	R31122Z1I - XJDE0002	R31122Z1I - XJDE0001		P31122Z1	R31122Z1	P311221
作業オーダー在庫出庫	F3111Z1	R31113Z1I - ZJDE0002	R31113Z1I - ZJDE0001		P3111Z1	R3111Z1P	該当なし
作業オーダー完了	F4801Z1	R31114Z1I - XJDE0002	R31114Z1I - XJDE0001		P4801Z1	R4801Z1P	該当なし
スーパー・バックフラッシュ	F3112Z1	R31123Z1I	R31123Z1I - ZJDE0001		P3112Z1	R3112Z1P	該当なし
部品表	F3002Z1	R3002Z1I - ZJDE0002	R3002Z1I - ZJDE0001		P3002Z1	R3002Z1P	P3002
作業工程マスター	F3003Z1	R3003Z1I - ZJDE0002	R3003Z1I - ZJDE0001		P3003Z1	R3003Z1P	P3003
作業場マスター	F30006Z1	R30006Z1I - ZJDE0002	R30006Z1I - ZJDE0001		P30006Z1	R30006Z1P	P3006
作業日カレンダー	F0007Z1	R0007Z1I - XJDE0002	R0007Z1I - XJDE0001		P0007Z1	R0007Z1P	P00071
計画メッセージ	F3411Z1	R3411Z1I - ZJDE0002	R3411Z1I - ZJDE0001		P3411Z1	R3411Z1P	該当なし
明細予測	F3460Z1	R3460Z1I - XJDE0002	R3460Z1I - XJDE0001		P3460Z1	R3460Z1P	P3460、R3465、R34650（個別に実行）
かんばんトランザクション	F30161Z1	R30161Z1I - XJDE0002	R30161Z1I - XJDE0001	該当なし	P30161Z1	R30161Z1P	該当なし

電子データ・インターフェイス(EDI)

電子データ交換システム用 J.D. Edwards データ・インターフェイスは、J.D. Edwards システム・データと変換ソフトウェア間のインターフェイスとして機能します。このデータ・インターフェイスは、EDI データ交換の他に、一般のインタオペラビリティにも使用されます。電子商取引では、ファイル基準のインターフェイスがビジネス要件を満たしている必要があります。

参照

- インタオペラビリティに電子データ交換(EDI)を使用する方法については、『Data Interface for Electronic Data Guide(電子データ用データ・インターフェイス)』ガイドの「Overview for Data Interface for Electronic Data Interchange System(電子データ交換システム用のデータ・インターフェイスの概要)」およびその他の項

テーブル変換

テーブル変換とは、テーブル内のデータを高速処理できる特殊な形式のユニバーサル・バッチ・エンジン(UBE)です。ERP には、企業データの収集、フォーマット、エクスポートおよびインポートに使用できるテーブル変換ユーティリティがあります。テーブル変換ツールを使用すると、データを転送、コピーしたり、テーブルからレコードを削除できます。テーブル変換により、J.D. Edwards 以外のテーブルを使用してビジネス関数を処理し、直接呼び出して出力できます。

参照

- テーブル変換については『テーブル・コンバージョン』ガイドの「テーブル変換の概要」およびその他の項

出力ストリーム・アクセス(OSA) UBE

OSA インターフェイスを設定している場合は、ERP データを他のソフトウェア・プログラムに渡して処理およびフォーマットさせることができます。OSA は専用のコマンド・セットを使用することも、XML ライブラリを使用することもできます。

参照

- OSA を使用して ERP データを他のソフトウェア・プログラムで処理する方法については、『エンタープライズ・レポート・ライティング』ガイドの「出力ストリーム・アクセス」

拡張プランニング・エージェント(APAg)/インテグレーション

J.D. Edwards の拡張プランニング・エージェント(APAg)は、企業データのバッチ抽出、変換、ロードを行うツールです。APAg は、リレーショナル・データベース、フラット・ファイル・フォーマットおよび XML のような他のデータまたはメッセージ・エンコーディング形式によるデータ・ソースへのアクセスをサポートしています。また、APAg はデータを 2 点間で移動し、そのデータ移動に関連するタスクを実行します。

参照

- APAg ツールについては、『Advanced Planning Agent 3_3_3 User Guide（拡張プランニング・エージェント 3_3_3 ユーザー・ガイド）』の「APAg Integrates Your System（APAg によるシステムの統合）」およびその他の項

オープン・データ・アクセス(ODA)

J.D. Edwards オープン・データ・アクセス(ODA) ODBC ドライバは、バージョン 2.5 以降に準拠した読取り専用ドライバです。フロント・エンドの Windows クエリーおよびレポート作成ツールでは、ODA を使用して J.D. Edwards ERP データベースにアクセスできます。サポートされているフロント・エンドのツールには、次のようなものがあります。

- Microsoft Query
- Microsoft Access
- Microsoft Excel
- ODBCTEST
- Crystal Report
- Microsoft Analysis Service（未認定）

ODA はフロントエンド・クエリー/レポート作成ツールと ERP が構成する ODBC ドライバの間に置かれます。

ERP データベースには、データが正しく表示されるように変換または適用する必要があるオブジェクトおよびカラム名、特定のデータ・タイプおよびセキュリティ・ルールがあります。特定のデータ・タイプおよびルールには、小数点シフト、ユリウス暦日付、通貨、メディア・オブジェクト、セキュリティ、およびユーザー定義コードなどがあります。一部のインスタンスでは、選択用ツール内でデータが正しく表示されるように、ODA がデータと同様に SQL SELECT ステートメントを修正することもあります。

ハードウェアおよびソフトウェアの要件

ODA を使用するには、次のハードウェア要件とソフトウェア要件を満たす必要があります。

ハードウェア要件

- IBM 互換のパーソナル・コンピュータ。
- 空きディスク容量が 6 MB あるハードディスク。
- 最低 16 MB のランダム・アクセス・メモリ(RAM)。

ソフトウェア要件

ODA ドライバを使用してデータにアクセスするには、システムが ERP の最低限必要な技術的条件(MTR)を満たしている必要があります。MTR はリリースごとに更新されて、オンラインで入手できます。また、次のソフトウェア要件も満たしている必要があります。

- J.D. Edwards ERP
- J.D. Edwards オープン・データ・アクセス・ドライバ(JDEOWODA.dll)
- 32 ビット ODBC ドライバ・マネージャ、バージョン 3.0 以降(ODBC32.dll)（このファイルは ODBC データベース・ドライバに付属）
- Microsoft Windows 95 以降または Windows NT 4.0 以降

Windows 95 で 16 ビット・アプリケーションによる ODA ODBC ドライバの使用はサポートされていません。

ODBC コンポーネントファイル

J.D. Edwards インストールは、ODBC データベース・ドライバが要求するコンポーネントをインストールします。この他に、通常は次の追加ファイルもインストールされます。

ドライバ	ファイル名
ODA ドライバ	JDEOWODA.DLL
ODA ドライバ・ヘルプ	JDEOWODA.HLP
リリース・ノート	README.TXT

注:

ERP 9.0 の場合、OLEDB は SQL Server 用の新規ドライバです。ただし、ODA では OLEDB データ・ソースはサポートされません。SQL Server で ODA を使用する場合は、ODBC を使用してデータ・ソースを設定してください。

オープン・データ・アクセス・ドライバのアーキテクチャ

J.D. Edwards ODA ODBC ドライバのアーキテクチャには 5 つのコンポーネントがあります。

- アプリケーション

  ODA ドライバを呼び出して J.D. Edwards データベースのデータにアクセスする、フロント・エンドのクエリー/レポート作成ツールです。

- マネージャ

  アプリケーションの代わりにドライバをロード、アンロードします。ODBC 呼出しを処理し、ODBC 呼出しをドライバに渡します。

- J.D. Edwards ODA ドライバ

  一部の ODBC 要求をベンダの ODBC ドライバに直接渡します。ERP 固有のデータ・タイプが使用される場合には、要求をベンダの ODBC ドライバに送信する前に SQL SELECT ステートメントが修正されます。ベンダの ODBC ドライバからデータが戻された後、J.D. Edwards ODA ODBC ドライバは、アプリケーション内でデータが正しく表示されるように、場合によってはデータを操作する必要があります。

- ベンダ・ドライバ

  ODBC ファンクション・コールを処理し、SQL 要求を特定のデータソースに投入します。必要があれば、関連 DBMS がサポートしている構文に要求が準拠するように、ドライバがアプリケーションの要求を修正します。

- データ・ソース

  アクセスしたいデータ、データ用のオペレーティング・システム、DBMS、およびネットワーク・プラットフォーム。

ODA データソースの追加

ODAドライバはインストール・プロセスの一部として自動的に登録されますが、場合によってはデータ・ソースを追加する必要があります。ファイル・データソースまたはシステム・データソースの追加、データソースの修正、または必要に応じてデータソースの削除を実行することもできます。また、場合によっては、ODAドライバを設定する際に現在値チェックボックスをチェックして、現在値が正しいフォーマットで表示されるようにする必要があります。

Oracle を使用する場合は、別の ODBC DSN を作成して OneWorld ODA Ora と命名し、ODA で Oracle データ・ソースにアクセスできるようにする必要があります。この処理に関する特定の情報は、オンライン・リリース・ノートに含まれます。

► データ・ソースを追加するには

1. [コントロール パネル]をダブルクリックします。[コントロール パネル]で、ODBC アイコンをダブルクリックします。
2. [ユーザー データ ソース]ダイアログ・ボックスで、[追加]をクリックします。
3. [データ・ソースの追加]で、[インストール済み ODBC ドライバ]リストから[J.D. Edwards Open Data Access driver]を選択して[完了]をクリックします。
4. [データソースの構成]に次の情報を入力してデータ・ソースを設定し、[OK]をクリックします。
   - データ・ソース名

     J.D. Edwards オープン・データ・アクセス・ドライバを呼び出す名前を指定します。

   - 記述

     追加するドライバの記述を指定します。記述は 79 文字以内で入力します。

5. 〈Connect(接続)〉フォームで、次のオプションのうち1つ以上をオンにします。
   - Convert User Defined Codes(ユーザー定義コードの変換)

     ユーザー定義コードの代わりにユーザー定義フィールドの関連記述を戻すには、このオプションを選びます。関連記述はよりわかり易い説明です。ユーザー定義コードに使用されるコードの代わりのテキスト説明になるためです。デフォルト値では、ユーザー定義コードの代わりに関連記述を表示します。

   - Convert Currency Values(通貨値の換算)

     通貨フィールドを正しい値に変換するには、このオプションを選びます。

   - Use Long Table/Business View Names(詳細(Long)テーブル/ビジネス・ビュー名)

     詳細なテーブル名/ビュー名を表示するには、このオプションを選びます。

   - Use Long Column Names(詳細(Long)カラム名)

     詳細なカラム名を表示するには、このオプションを選びます。

6. 〈Connect〉フォームで、次のテーブル/ビジネス・ビュー表示オプションのうち1つ以上をオンにします。

   - Tables Only（テーブルのみ）

     J.D. Edwards ERP テーブルのみを表示するには、このオプションを選びます。

   - Business Views Only（ビジネス・ビューのみ）

     J.D. Edwards ERP ビジネス・ビューのみを表示するには、このオプションを選びます。

   - Tables and Business Views（テーブルおよびビジネス・ビュー）

     J.D. Edwards ERP テーブルと J.D. Edwards ERP ビジネス・ビューの両方を表示するには、このオプションを選びます。

7. ODA で使用可能な関数のリストをカスタマイズするには、[Advanced（上級）]をクリックします。

   上級コンフィギュレーションはオプションです。ODA で使用可能な関数のリストをカスタマイズしない場合は、デフォルト・リストの設定が使用されます。

ファイル・データソースの追加

また、ファイル・データソースを追加することもできます。

▶ ファイル・データソースを追加するには

1. [コントロール パネル]をダブルクリックします。[コントロール パネル]で、ODBC アイコンをダブルクリックします。
2. [ユーザー データ ソース]で、[DSN]タブをクリックします。
3. [ファイル データ ソース]で、[Add]をクリックします。
4. [データ・ソースの追加]で、[インストール済み ODBC ドライバ]リストから[J.D. Edwards Open Data Access driver]を選択して[完了]をクリックします。
5. [データソースの構成]に次の情報を入力してデータ・ソースを設定し、[OK]をクリックします。

   - データ・ソース名

     J.D. Edwards オープン・データ・アクセス・ドライバを呼び出す名前を指定します。

   - 記述

     追加するドライバの記述を指定します。記述は 79 文字以内で入力します。

6. 〈Connect〉フォームで、次のオプションのうち1つ以上をオンにします。

   - Convert User Defined Codes（ユーザー定義コードの変換）

     ユーザー定義コードの代わりにユーザー定義フィールドの関連記述を戻すには、このオプションを選びます。関連記述はよりわかり易い説明です。ユーザー定義コードに使用されるコードの代わりのテキスト説明になるためです。デフォルト値では、ユーザー定義コードの代わりに関連記述を表示します。

   - Convert Currency Values（通貨値の変換）。

通貨フィールドを正しい値に変換するには、このオプションを選びます。

- Use Long Table/Business View Names

  詳細なテーブル名/ビュー名を表示するには、このオプションを選びます。

- Use Long Column Names

  詳細なカラム名を表示するには、このオプションを選びます。

7. 〈Connect〉フォームで、次のテーブル/ビジネス・ビュー表示オプションのうち1つ以上をオンにします。

   - Tables Only

     J.D. Edwards テーブルのみを表示するには、このオプションを選びます。

   - Business Views Only

     J.D. Edwards ビジネス・ビューのみを表示するには、このオプションを選びます。

   - Tables and Business Views

     J.D. Edwards テーブルと J.D. Edwards ビジネス・ビューの両方を表示するには、このオプションを選びます。

8. ODA で使用可能な関数のリストをカスタマイズするには、[Advanced(上級)]をクリックします。

   上級コンフィギュレーションはオプションです。ODA で使用可能な関数のリストをカスタマイズしない場合は、デフォルト・リストの設定が使用されます。

システム・データソースの追加

データソースは、同じマシン上の複数のユーザーが使用できるシステム・データソース名(DSN)を使用して設定できます。システム DSN は、システム規模のサービスからも使用可能です。このようなサービスからは、マシンにログオンしているユーザーがいない場合でもデータ・ソースへのアクセスが可能です。

▶ システム・データソースを追加するには

1. [コントロール パネル]をダブルクリックします。[コントロール パネル]で、ODBC アイコンをダブルクリックします。
2. [データ ソース]で、[システム DSN]タブをクリックして、[Add]をクリックします。
3. [システム データ ソース]で、[Add]をクリックします。
4. [Add Data Source(データ・ソースの追加)]で、[Installed ODBC Drivers(インストール済み ODBC ドライバ)]リストから[J.D. Edwards Open Data Access driver]を選択して[Finish(完了)]をクリックします。
5. [Configure Data Source(データソースの構成)]に次の情報を入力してデータ・ソースを設定し、[OK]をクリックします。

   - データソース名 – J.D. Edwards オープン・データ・アクセス・ドライバを呼び出す名前を指定します。

- 記述 – 追加しているドライバの記述を指定します。(記述の入力は 79 文字を超えてはならないことに注意してください。)

6. 〈Connect(接続)〉フォームで、次のオプションのうち1つ以上を使用可能にします。
   - ユーザー定義コード変換 – ユーザー定義コードの代わりにユーザー定義フィールドの関連記述を戻すために、このオプションを使用します。関連記述はよりわかり易い説明です。ユーザー定義コードに使用されるコードの代わりのテキスト説明になるためです。デフォルトでは、ユーザー定義コードの代わりに関連記述を表示します。
   - 通貨値の換算 – 通貨フィールドを正しい値に変換するために、このオプションを使用します。
   - 詳細テーブル/ビジネス・ビュー名の使用 – 詳細テーブル/ビュー名を表示するために、このオプションを使用します。
   - 詳細カラム名の使用 – 詳細カラム名を表示するために、このオプションを使用します。
7. 〈Connect〉フォームで、次のテーブル/ビジネス・ビュー表示オプションのうち1つまたは複数を使用可能にします。
   - テーブルのみ – J.D. Edwards ERP テーブルのみを表示するために、このオプションを使用します。
   - ビジネス・ビューのみ – J.D. Edwards ERP ビジネス・ビューのみを表示するために、このオプションを使用します。
   - テーブルおよびビジネス・ビュー – J.D. Edwards ERP テーブルと J.D. Edwards ERP ビジネス・ビューの両方を表示するために、このオプションを使用します。
8. ODA で使用可能な関数のリストをカスタマイズするには、[Advanced(上級)]をクリックします。

   上級コンフィギュレーションはオプションです。ODA で使用可能な関数のリストをカスタマイズしない場合は、デフォルト・リストの設定が使用されます。

データソースの修正

また、データソースを変更することもできます。

► データソースを変更するには

1. [コントロール パネル]をダブルクリックします。[コントロール パネル]で、ODBC アイコンをダブルクリックします。
2. [ユーザー データ ソース]、[ファイル データ ソース]、または[システム データ ソース]で、使用可能なリストからデータ・ソースを選択します。
3. [構成]をクリックします。
4. [Configure Data Source(データ・ソースの構成)]に次の情報を入力してデータ・ソースを設定し、[OK]をクリックします。
   - Data Source Name(データ・ソース名)

     J.D. Edwards オープン・データ・アクセス・ドライバを呼び出す名前を指定します。
   - Description(記述)

     追加するドライバの記述を指定します。記述は 79 文字以内で入力します。

5. 〈Connect〉フォームで、次のオプションのうち1つ以上をオンにします。

   - Convert User Defined Codes（ユーザー定義コードの変換）

     ユーザー定義コードの代わりにユーザー定義フィールドの関連記述を戻すには、このオプションを選びます。関連記述はよりわかり易い説明です。ユーザー定義コードに使用されるコードの代わりのテキスト説明になるためです。デフォルトでは、ユーザー定義コードの代わりに関連記述を表示します。

   - Convert Currency Values（通貨値の変換

     通貨フィールドを正しい値に変換するには、このオプションを選びます。

   - Use Long Table/Business View Names

     詳細なテーブル名/ビュー名を表示するには、このオプションを選びます。

   - Use Long Column Names

     詳細なカラム名を表示するには、このオプションを選びます。

6. 〈Connect〉フォームで、次のテーブル/ビジネス・ビュー表示オプションのうち1つ以上をオンにします。

   - Tables Only

     J.D. Edwards ERP テーブルのみを表示するには、このオプションを選びます。

   - Business Views Only

     J.D. Edwards ERP ビジネス・ビューのみを表示するには、このオプションを選びます。

   - Tables and Business Views

     J.D. Edwards ERP テーブルと J.D. Edwards ERP ビジネス・ビューの両方を表示するには、このオプションを選びます。

7. ODA で使用可能な関数のリストをカスタマイズするには、[Advanced（上級）]をクリックします。

   上級コンフィギュレーションはオプションです。ODA で使用可能な関数のリストをカスタマイズしない場合は、デフォルト・リストの設定が使用されます。

データソースの削除

データソースを削除することもできます。

▶ データソースを削除するには

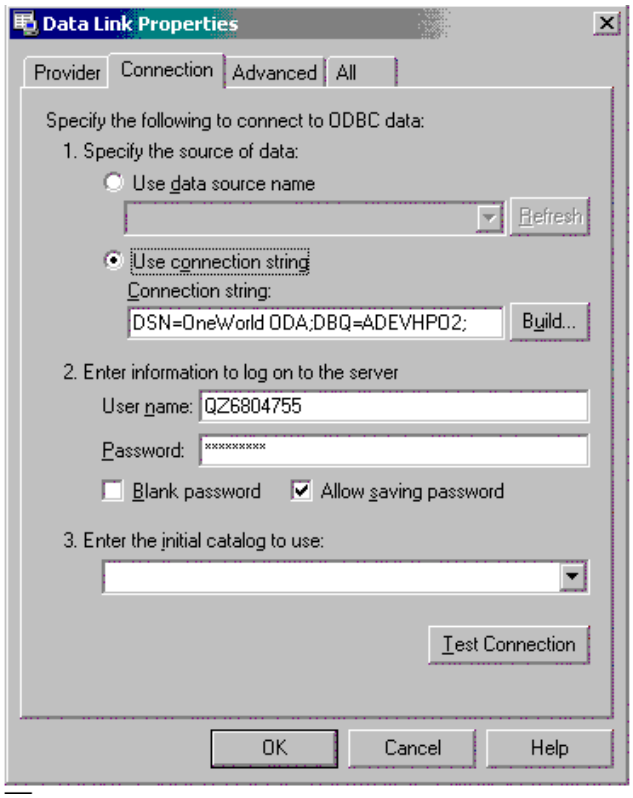
1. [コントロール パネル]をダブルクリックします。[コントロール パネル]で、ODBC アイコンをダブルクリックします。
2. [ユーザー データ ソース]、[ファイル データ ソース]、または[システム データ ソース]で、[データ ソース]リストから削除したいデータ・ソースを選択します。
3. [削除]をクリックしてから[Yes]をクリックし、削除を確認します。

接続文字列のキーワードを使用する

C プログラミング言語を使用して、SQLDriverConnect や SQLBrowseConnect など、ODA にサポートされている SQL API を直接呼び出すデータベース・アプリケーションを記述できます。次の表に、独自のデータベース・アプリケーションを記述する際に接続文字列の中で使用するキーワードを示します。

キー	値	説明	入力接続文字列	出力接続文字列
CONVERTUDC	Y/N （デフォルトは N）	UDC の変換 （Y/N）	必要に応じて入力します。接続文字列に含まれない場合は、INI/レジストリ設定（ERP ODA DSN 設定）からロード。	入力文字列または INI/レジストリ設定から
CONVERTCURRENCY	Y/N （デフォルトは N）	通貨の変換 (Y/N)		
SHIFTDECIMALS	Y/N （デフォルトは Y）	小数点シフトの使用(Y/N)		
CONVERTJULIANDATES	Y/N （デフォルトは Y）	ユリウス暦の変換(Y/N)		
DISPLAYOPTIONS	0/1/2 （デフォルトなし）	TBLE、BSFN の表示（一方または両方）		
LONGTABLENAMES	Y/N （デフォルトは Y）	テーブルに詳細名称を使用 (Y/N)		
LONGCOLUMNNAMES	Y/N （デフォルトは Y）	カラム名に詳細名称を使用(Y/N)		
UID	<文字列>	ユーザーID	JDEDriverConnect (SQL_DRIVER_NOPROMPT)により必須	OW ログインによって上書きされない場合は入力内容に同じ
PWD	<文字列>	パスワード		
ENVIRONMENT	<文字列>	環境		
DBQ	<文字列>	ENVIRONMENT に同じ	[ENVIRONMENT] として機能。指定しない場合は [ENVIRONMENT] になります。	ENVIRONMENT 値が存在する場合は除去されます。
DSN	<文字列>	データ・ソース	任意。無効な場合のデフォルトは [DEFAULT]	ログイン情報により上書きされます。

Microsoft Analysis Service ツールを使用する場合は、接続文字列キーワードを使って新しいデータ・ソースを作成できます。次の画面に、Microsoft Analysis Service ツールで接続文字列キーワードを使用する方法を示します。



ODA の処理

ODA ドライバが正常にインストールされ、ODBC データ・ソースが確立されると、ODA ドライバの機能を使用できます。SQL 接続が確立されると、現在の接続の環境がデータベース名としてシステムに格納されます。この値は後で SQLGetInfo がアクセスしたり、将来の接続に使用したりできます。

J.D. Edwards ODA と共に、次の J.D. Edwards 固有の機能を使用できます。

詳細(Long)テーブルおよびビジネス・ビュー名

詳細テーブルおよびビジネス・ビュー名を使用すると、オブジェクト・リストを表示する場合に内容を把握しやすい名前で処理できます。SELECT ステートメントではこれらの名称、または元々の J.D. Edwards オブジェクト名のいずれかを使用できます。

注:

サードパーティ製品の中には、リリース 2.0 以前の ShowCase STRATEGY 製品や Crystal Reports のように、このオプションが使用できないものもあります。詳細名にはこれらのツールで正しく取り扱われない特殊文字が含まれるからです。

詳細(Long)カラム名

詳細カラム名を使用すると、カラム・リストを表示する場合に内容を把握し易い名称で処理できます。これらの名称、または元々の J.D. Edwards カラム名いずれかを使用できます。たとえば、次のいず

れかのステートメントを使用して、住所録マスター・テーブル(F0101)からデータを取り込むことができます。

- SELECT ABAN8 from the Address Book Master table (F0101)
- SELECT AddressNumber from the Address Book Master table (F0101)

ユリウス暦日付

ユリウス暦日付は、すべての参照日付を SQL-92 標準日付に変換するために、ユリウス暦日付カラムに修正します。J.D. Edwards ユリウス暦日付は、日付計算に使用できる標準日付値に変換されます。この機能を使用すると、SELECT(結果データ)、WHERE、および HAVING 句と ORDER BY 句の両方で、期間および他の日付計算を使用できます。

SQL SELECT ステートメントは、データ計算の前に J.D. Edwards ユリウス暦日付カラムから標準日付に変換するために修正されます。SQL SELECT ステートメントの修正は、日付計算処理のドライバの違いのため、アクセスされるデータ・ソースに基づいています。当初カラム値が 0(ゼロ)の場合、日付変換の結果は日付値 1899-12-31 となります。これらの値を削除するには、SELECT ステートメントの WEHERE 句に次の条件を追加する必要があります。この場合、DATECOL は J.D. Edwards ユリウス暦日付カラムです。

```
"DATECOL <> {d `1899-12-31'}"
```

小数点シフト

小数点シフト・カラムに対するすべての参照は、結果データが正しくなるように小数点がシフトするために修正されます。この機能により、複雑な計算式、総計、フィルタリングを含む SQL ステートメントを実行し、正確な結果を戻すことができます。

SQL SELECT ステートメントが修正され、データが正しく戻るように、適切な小数部の桁数別にカラムが分けられ、比較演算子がフィルタリング処理をできるように修正します。

通貨

通貨カラムは、選択されたカラム・リスト内で単一カラム参照に制限されます。戻されるデータは J.D. Edwards 標準通貨変換ルーチンを使用して変換されています。SQL ステートメントでの通貨カラムに対するその他の参照はすべてネイティブ・ドライバに渡されます。フィルタリングを有効利用するためには、通貨カラムの使用方法を理解する必要があります。

選択されたカラムが戻される前に、J.D. Edwards オープン・データ・アクセス・ドライバがすべての通貨カラムを正しい値に変換します。WHERE 句または HAVING 句に使用される通貨カラムは、未変換の通貨値に基づいて処理されます。GROUP BY 句または ORDER BY 句の通貨カラムは未変換通貨値によってグループ化され、ソートされます。

メディア・オブジェクト

メディア・オブジェクト(F00165)ストレージに含まれるメディア・オブジェクト・カラム、TXVC は、選択されたカラム・リスト内で単一カラム参照に制限されます。ODA はメディア・データを、プレーン・テキストまたはリッチ・テキスト形式(RTF)で返し、画像などその他のバイナリ・データは切り捨てます。テキストまたは RTF のサイズは 30,000 文字に制限されており、この制限を超えたテキストは切り捨てられます。

カラムセキュリティ

カラム・セキュリティがアクティブな場合に、制限されたカラムへの参照があると、SELECT ステートメントの検証時にエラーが戻されます。SQL-02 標準で定義されているように、SELECT 句に*(アスタ

リスク – すべてのカラムを選択）が使用されている場合などです。テーブルのすべてのカラムに対して権限がない場合は、エラーを受け取ることになります。

ロー・セキュリティ

ロー・セキュリティがアクティブの場合、セキュリティのかかったローにフィルタリングをかけるため、構文が修正されて適切な WHERE 句が組み込まれます。表示されるのはアクセスを許可されているローのみであり、それと同時に、SUM や AVG などの集計関数の使用により正確な結果が得られます。

ユーザー定義コード

ユーザー定義コード(UDC)が有効な場合、カラム・データが戻される場合に内部コードの代わりに関連記述が表示されます。この処理は、戻されるデータだけに影響があり、SELECT ステートメントの他の部分、たとえば WHERE、ORDER BY 句には影響がありません。これは、ドライバの設定時に構成できる任意の設定です。

UDC がユーザーに戻される前に、J.D. Edwards オープン・データ・アクセス・ドライバがコードを関連記述に変換します。WHERE 句または HAVING 句に使われる UDC カラムは、未変換コードに基づいて選択され、GROUP BY 句および ORDER BY 句で参照される UDC カラムは未変換コード別にグループ化され、ソートされます。

Microsoft Excel を使用してクエリーを実行する

Microsoft Excel を使用してクエリーを作成し、実行できます。

► Microsoft Excel を使用してクエリーを実行するには

1. [データ]メニューから、[外部データの取り込み]を選びます。
2. [新しいクエリーの作成]を選びます。
3. [データベース]タブで、該当するデータ・ソース（ERP Local、ERP ODA など）を選びます。

   Excel はファイル・データソースを使用するので、32 ビット ODBC 管理者で設定した ODA データソースはデータベースリストに表示されません。〈新規データ・ソース〉を選んでファイル・タイプ・データ・ソースを作成し、データ・ソースの設定手順に従ってください。

   ODA データソースを選択する場合は、ERP にログオンして ODA ドライバを使用する必要があります。ログオンすると、Solution Explorer が表示されません。ODA ドライバがセキュリティおよび環境マッピングをチェックできるようにアクティブになっているにすぎないからです。

   Excel クエリー・ウィザードにより、ERP データソースの使用可能なテーブル・リストが表示されます。1つのテーブルを展開表示すると、各テーブルの使用可能なカラムまたはフィールドが表示されます。ODA ドライバを使用している場合は、各フィールドの詳細記述（DateUpdated など）が表示されます。ODA ドライバを使用していない場合は、フィールドの名称記述コード（ABUPMJ など）が表示されます。

4. フィールド/カラム名を J.D. Edwards 名称コードから変換するには、データ・フィールド表示テキスト・テーブル(F9202)を使用します。すべてのローを選び、FRDTAI でソートして、相互参照を作成します。

   すべての J.D. Edwards カラム名の最初の 2 文字はアプリケーションコードで、残りの文字は、サフィックスとしてこのテーブル内に置かれます。

5. クエリー・ウィザードを使用してクエリーの作成を終了し、クエリーを保存します。
6. 次にクエリーを実行して Excel または MS Query でクエリーを検討できます。

   Excel からクエリーを実行した後、MS Query での結果を見たい場合には、クエリーはページ単位で選択するので、直ちに結果を戻すことができます。大量の結果を処理している場合は、レコード間をすばやく移動できるように、大量なメモリを使用するアプリケーションや ERP を閉じる必要があります。MS Query ではなくスプレッドシートに直接クエリーを戻させたい場合は、どのローであっても表示するには時間がかかることがあります。Excel はページ単位で表示しないためです。

各クエリーの結果を検証するには、ODA 以外の ERP データソースを使用してまず各クエリーを実行し、次に ODA データソースを使用して結果を比較します。

ODA エラーメッセージ

J.D. Edwards オープン・データ・アクセス・ドライバから受信するエラーのリストを次に示します。メッセージは、アプリケーションが ODBC 標準エラーメカニズムを使用して受信できる ODBC エラーメッセージ待ち行列にメッセージが入れられます。J.D. Edwards メッセージは次のとおりです。

[J.D. Edwards][OneWorldODA Driver]MESSAGE TEXT

Configuration Request Error(構成要求エラー)

ドライバに充分なデータを提供しておらず、[構成]ダイアログを表示できない時に新規データ・ソースを追加すると、このエラーが発生します。

ドライバに充分なデータを渡すか、またはドライバが詳細データを求めてプロンプトを出すことができるようにする必要があります。

Option Value Changed(オプション値が変更されました)

これは、接続オプションまたはステートメント・オプションをそのドライバが受け入れない値に設定しようとする時に発生する情報メッセージです。ドライバが受入れ可能なデフォルト値に値を変更し、ユーザーに値が変更されたことを通知するためにこのメッセージを使用します。

J.D. Edwards オープン・データ・アクセス・ドライバは次の領域の値を変更します。

- ローセット・サイズの1以外の数への設定。ドライバは現在単一ローのローセットしかサポートしていません。
- ログインのタイム・アウトを 0 以外の値に設定します。現在、ドライバはこのオプションでゼロのみをサポートしています。つまり、タイムアウトは使用不可です。

Data Source Name Is Not Valid(データソース名が無効です)

入力したデータソースが有効な ODBC データソース名ではありません。新規のデータソースを追加するかまたは既存のデータソースを構成する場合に、このエラーが発生します。ODBC データソース命名規則に従った名前を入力する必要があります。

Data Source Does Not Exist(データソースが存在しません)

存在しないデータソースを使用しようとすると、このエラーが発生します。既存のデータソースの名前を入力する必要があります。あるデータ・ソースに接続しようとする際にこのエラーが発生した場合は、デフォルトのデータ・ソースを作成する必要があります。

Unable to Allocate Memory(メモリを割り当てできません)

J.D. Edwards オープン・データ・アクセス・ドライバは継続するための充分なメモリを割り当てることができませんでした。一部のアプリケーションを閉じて、もう一度操作を実行する必要があります。最小システム要件に適合しているか確認してください。

Invalid Type of Request(要求タイプが無効です)

ドライバが認識しない構成オプションの使用を試みました。データソースの構成時には、ドライバは次のオプションをサポートしています。

- データソースの追加
- データソースの構成
- データソースの削除

Data Truncated(データが切り捨てられました)

カラム・データの変換の結果、値が切り捨てられました。この情報メッセージを出されないようにするには、カラム・データ幅を広げる必要があります。

Syntax Error or Access Violation(構文エラーまたはアクセス違反)

ステートメントに構文エラーがあり、データを使用できません。

Unable to Display Connection Dialog([Connection(接続)]ダイアログを表示できません)

[Connection]ダイアログの表示を試みた時にドライバがこのエラーを検出しました。

Cross System Joins Not Supported(システム間結合はサポートされていません)

このエラーは次の 2 つの状況のいずれかで発生します。

- ERP 環境で複数のシステム上に含まれているテーブルを参照しました。J.D. Edwards オープン・データ・アクセス・ドライバは、現在単一システム上で参照されるテーブルをサポートしています。
- 複数のシステムに常駐する複数のテーブルを含むビジネス・ビューを参照しました。

単一システム上でテーブル、または単一システム上で複数テーブルが含まれるビジネス・ビューを参照しているかを確認する必要があります。

Unable to Connect to the ERP Environment（ERP 環境に接続できません）

ドライバは J.D. Edwards ERP 環境への接続を確立できませんでした。このドライバへの接続が正常に終了する前に接続が要求されました。

Internal Data Conversion Error（内部データ変換エラー）

データ変換中に、ドライバが未確認エラーを検出しました。

Internal Execution Error（内部実行エラー）

ステートメント実行中に、ドライバが予期しないエラーを検出しました。

User Defined Code Columns Can Only Be in Simple Column References（ユーザー定義コードは、簡単なカラム参照にしか入れることができません）

ユーザー定義コードを複雑な計算式に使用しようとしました。J.D. Edwards オープン・データ・アクセス・ドライバでは、そのようなカラムは簡単な参照にしか使用できません。

Currency Columns Can Only Be in Simple Column References（通貨カラムは簡単なカラム参照にしか入れることができません）

通貨カラムを複雑な計算式に使用しようとしました。J.D. Edwards オープン・データ・アクセス・ドライバでは、そのようなカラムは簡単な参照にしか使用できません。

Media Object Columns Can Only Be in Simple Column References（メディア・オブジェクト・カラムは簡単なカラム参照にしか入れることができません）

メディア・オブジェクト・カラムを複雑な計算式に使用しようとしました。J.D. Edwards オープン・データ・アクセス・ドライバでは、そのようなカラムは簡単な参照にしか使用できません。

Column Security Violation（カラム・セキュリティ違反）

使用権限がないカラムを使用しようとしました。セキュリティがかかっているカラムに対する参照を削除する必要があります。

Invalid Cursor State（カーソル状態が無効です）

ドライバの置かれている状態で、無効な操作を試みました。たとえば次のとおりです。

- ステートメントを準備、または実行する前にカラムをバインドしようとしました。
- 保留結果がある最中にステートメントを実行しようとしました。
- ステートメントを準備、または実行する前にドライバからデータを取得しようとしました。
- 保留結果がある最中に、ステートメントを準備しようとしました。

Invalid Column Number(カラム番号が無効です)

ステートメントの結果の一部ではないカラムにアクセスしようとしました。

Driver Does Not Support the Requested Conversion(ドライバは要求された変換をサポートしていません)

J.D. Edwards オープン・データ・アクセス・ドライバがサポートしていないデータ・タイプにカラムを変換しようとしました。

Invalid Date/Time String(日付／時刻文字列が無効です)

文字カラムが無効なフォーマットであったため、文字カラムの日付、時刻またはタイムスタンプ値への変換が失敗しました。

Invalid Numeric String(数値文字列が無効です)

文字カラムに無効な数値があるため、文字カラムの数値への変換が失敗しました。

Numeric Value Out of Range(数値が範囲外です)

出力データ・タイプがカラムの値に適応しなかったため、文字カラムの数値への変換が失敗しました。デフォルトのデータ・タイプを使用するか、カラム値に適応するデータ・タイプを選択する必要があります。

Data Returned for One or More Columns was Truncated(1つ以上のカラムに戻されたデータが切り捨てられました)

カラムを数値に変換しようとした結果、小数点桁数が切り捨てられました。出力データ・タイプがカラムの値に適応できませんでした。デフォルトのデータ・タイプを使用するか、カラム値に適応するデータ・タイプを選択する必要があります。

The Data Cannot be Converted(データを変換できません)

入力タイプを出力タイプに変換できなかったので、カラム値の変換が失敗しました。デフォルトのデータ・タイプを使用する必要があります。

Statement Must Be a SELECT(ステートメントは SELECT でなくてはなりません)

J.D. Edwards オープン・データ・アクセス・ドライバは読取り専用で、SELECT ステートメントのみを使用できます。

Attempt to Fetch Before the First Row(最初のローの前にフェッチを試みました)

結果の開始前に、フェッチを試みました。この試みの結果、最初のローセットがフェッチされました。

Option Value Changed(オプション値が変更されました)

接続、ステートメントまたはスクロールを許可されていない値をセットしようとしました。J.D. Edwards オープン・データ・アクセス・ドライバが類似した値に置換しました。

Fractional Truncation(部分トランザクション)

出力データ・タイプがカラムの値に適応しなかったため、カラムを数値に変換する試みが正常に終了し、わずかな桁数が失われました。デフォルトのデータ・タイプを使用するか、カラム値に適応するデータ・タイプを選択する必要があります。

Driver Not Capable(ドライバが対応できません)

ドライバが対応できない接続、ステートメントまたはスクロールオプションをセットしようとしました。

Multiple Business Views Referenced(複数のビジネス・ビューを参照しました)

単一の SELECT 文で、複数のビジネス・ビューを参照しようとしました。J.D. Edwards オープン・データ・アクセス・ドライバは、1つのビジネス・ビューのみを含めるように SELECT ステートメントを制限しています。

Unable to Open Table or Business View(オープンテーブルまたはビジネス・ビューを使用できません)

J.D. Edwards ERP オープン・データ・アクセス・ドライバが ERP データベースにテーブルまたはビジネス・ビューを見つけることができなかったか、またはテーブルまたはビジネス・ビューに関係するデータを取得できませんでした。

Server Connection Failed(サーバー接続が失敗しました)

J.D. Edwards オープン・データ・アクセス・ドライバが SELECT ステートメントのテーブルまたはビジネス・ビューによって参照されるサーバーへの接続を確立できませんでした。

Business View Contains Invalid Join(ビジネス・ビューに無効な結合が含まれています)

ビジネス・ビュー定義に J.D. Edwards オープン・データ・アクセス・ドライバによって処理できない結合条件が含まれています。

Business View Contains Unsupported UNION Operator(ビジネス・ビューにサポートされていない UNION 演算子が含まれています)

ビジネス・ビュー定義に J.D. Edwards オープン・データ・アクセス・ドライバによって処理できない UNION 演算子が含まれています。

XMLフォーマット例（すべてのパラメータ）

次の例は、特定のフォーマットに使用します。

例：受信受注オーダーXML フォーマット（すべてのパラメータ）

この例は、パラメータがすべてある受信 XML フォーマットを示しています。

```
"<?xml version='1.0' ?>
<jdeRequest type='callmethod' user='userid' pwd='password' environment='environment'
>
<callMethod name=' GetLocalComputerId' app='NetCommerce' runOnError='no'>
<params>
<param name=' szMachineKey'id='m2' >< /param>
<params>
    <callMethod>
<callMethod name=' F4211FSBeginDoc' app='NetCommerce' runOnError='no'>
<params>
  <param name='mnCMJobNumber` id='j1`></param>
<param name='cCMDocAction' >A</param>
<param name='cCMProcessEdits' <1>/param>
<param name='szCMComputerID' idref='c2' >< /param>
<param name='cCMErrorConditions' >value< /param>
<param name='cCMUpdateWriteToWF' >value< /param>
   <param name='szCMProgramID' >value< /param>
   <param name='szCMVersion' >value< /param>
<param name='szOrderCo' <value< /param>
<param name='mnOrderNo' >value< /param>
<param name='szOrderType' >value< /param>
<param name='szBusinessUnit' >value< /param>
<param name='szOriginalOrderCo' >value< /param>
<param name='szOriginalOrderNo' >value< /param>
<param name='szOriginalOrderType' >value< /param>
```

```
<param name='mnAddressNumber' >value< /param>
<param name='mnShipToNo' >value< /param>
<param name='jdRequestedDate' >value< /param>
<param name='jdOrderDate' >value< /param>
   <param name='jdPromisedDate' >value< /param>
<param name='jdCancelDate' >value< /param>
<param name='szReference' >value< /param>
<param name='szDeliveryInstructions1' >value< /param>
   <param name='szDeliveryInstructions2' >value< /param>
<param name='szPrintMesg' >value< /param>
   <param name='szPaymentTerm' >value< /param>
<param name='cPaymentInstrument' >value< /param>
   <param name='szAdjustmentSchedule' >value< /param>
   <param name='mnTradeDiscount' >value< /param>
   <param name='szTaxExplanationCode' >value< /param>
   <param name='szTaxArea' >value< /param>
   <param name='szCertificate' >value< /param>
   <param name='cAssociatedText' >value< /param>
   <param name='szHoldOrdersCode' >value< /param>
<param name='cPricePickListYN' >value< /param>
<param name='mnInvoiceCopies' >value< /param>
<param name='mnBuyerNumber'>value</param>
<param name='mnCarrier' >value</param>
<param name='szRouteCode' >value</param>
<param name='szStopCode' >value</param>
<param name='szZoneNumber' >value</param>
<param name='szFreightHandlingCode' >value</param>
<param name='cApplyFreightYN' >value</param>
<param name='mnCommissionCode1' >value</param>
<param name='mnCommissionRate1' >value</param>
<param name='mnCommissionCode2' >value</param>
```

<param name='mnCommissionRate2' >value</param>
<param name='szWeightDisplayUOM' >value</param>
<param name='szVolumeDisplayUOM' >value</param>
<param name='szAuthorizationNo' >value</param>
<param name='szCreditBankAcctNo' >value</param>
<param name='jdCreditBankExpiredDate' >value</param>
<param name='cMode' >value</param>
<param name='szCurrencyCode' >value</param>
<param name='mnExchangeRate' >value</param>
<param name='szOrderedBy' >value</param>
<param name='szOrderTakenBy' >value</param>
<param name='szUserReservedCode' >value</param>
<param name='jdUserReservedDate' >value</param>
<param name='mnUserReservedAmnt' >value</param>
<param name='mnUserReservedNo' >value</param>
<param name='szUserReservedRef' >value</param>
<param name='jdDateUpdated' >value</param>
<param name='szUserID' >value</param>
<param name='szWKBaseCurrency' >value</param>
<param name='cWKAdvancedPricingYN' >value</param>
<param name='szWKCreditMesg' >value</param>
<param name='szWKTempCreditMesg' >value</param>
<param name='cWKInvalidSalesOrderNo' >value</param>
<param name='cWKSourceOfData' >blank</param>
<param name='cWKProcMode' >blank</param>
<param name='mnWKSuppressProcess' >0</param>
<param name='mnSODDocNo' >value</param>
<param name='szSODDocType' >value</param>
<param name='szSODOrderCo' >value</param>
<param name='mnTriangulationRateFrom' >value</param>
<param name='mnTriangulationRateTo' >value</param>

```
   <param name='cCurrencyConversionMethod' >value</param>
   <param name='cRetrieveOrderNo' >value</param>
   <param name='szPricingGroup' >value</param>
<param name='cCommitInvInED' >value</param>
<param name='cSpotRateAllowed' >value</param>
<param name='cGenericChar2_EV02' >value</param>
   <param name='szGenericString1_DL01' >value</param>
<param name='szGenericString2_DL02' >value</param>
<param name='mnGenericMathNumeric1_MATH01' >value</param>
<param name='mnGenericMathNumeric2_MATH02' >value</param>
<param name='szLongAddressNumberShipto' >value</param>
   <param name='szLongAddressNumber' >value</param>
   <param name='mnProcessID' >value</param>
<param name='mnTransactionID' >value</param>
</params>
<onError abort='yes'>\
<callMethod name=' F4211ClearWorkFile'app='NetCommerce' runOnError='yes'>
<params>
  <param name='mnJobNo` idref='j1`></param>
<param name='szComputerID' idref='c2' >< /param>
 <param name='mnFromLineNo' >value</param>
<param name='mnThruLineNo' >value</param>
<param name='cClearHeaderWF' >value</param>
<param name='cClearDetailWF' >value</param>
<param name='szProgramID' >value</param>
<param name='mnWKRelatedOrderProcess' >value</param>
<param name='szCMVersion' >value</param>
<param name='cGenericChar1_EV01' >value</param>
<param name='szGenericString1_DL01' >value</param>
    <param name='mnSODRelatedJobNumber' >value</param>
<param name='mnProcessID' >value</param>
```

```
<param name='mnTransactionID' >value</param>
</params>
</callMethod>
</onError>
</callMethod>
<callMethod name=' F4211FSEditLine'app='NetCommerce' runOnError='yes'>
<params>
<param name='mnCMJobNo ` idref='j1`></param>
<param name='cCMLineAction' >value</param>
<param name='cCMProcessEdits ' >value</param>
<param name='cCMWriteToWFFlag ' >value</param>
<param name='cCMRecdWrittenToWF ' >value</param>
<param name='szCMComputerID' idref='c2' >< /param>
<param name='cCMErrorConditions' >value</param>
<param name='szOrderCo' >value</param>
<param name='mnOrderNo' >value</param>
<param name='szOrderType' >value</param>
<param name='mnLineNo' >value</param>
<param name='szBusinessUnit' >value</param>
<param name='mnShipToNo' >value</param>
<param name='jdRequestedDate' >value</param>
<param name='jdPromisedDate' >value</param>
<param name='jdCancelDate' >value</param>
<param name='jdPromisedDlvryDate' >value</param>
<param name='szItemNo' >value</param>
<param name='szLocation'>value</param>
<param name='szLotNo' >value</param>
<param name='szDescription1' >value</param>
<param name='szDescription2' >value</param>
<param name='szLineType' >value</param>
<param name='szLastStatus' >value</param>
```

```
<param name='szNextStatus' >value</param>
<param name='mnQtyOrdered' >value</param>
<param name='mnQtyShipped' >value</param>
<param name='mnQtyBackordered' >value</param>
<param name='mnQtyCanceled' >value</param>
<param name='mnExtendedPrice' >value</param>
<param name='mnExtendedCost' >value</param>
<param name='szPrintMesg' >value</param>
<param name='cPaymentInstrument' >value</param>
<param name='szAdjustmentSchedule' >value</param>
<param name='cSalesTaxableYN' >value</param>
<param name='cAssociatedText' >value</param>
<param name='szTransactionUOM' >value</param>
<param name='szPricingUOM' >value</param>
<param name='mnItemWeight' >value</param>
<param name='szWeightUOM' >value</param>
<param name='mnForeignUnitPrice >value</param>
<param name='mnForeignExtPrice' >value</param>
<param name='mnForeignUnitCost' >value</param>
<param name='mnForeignExtCost' >value</param>
<param name='szPricingCategoryLevel' >value</param>
<param name='mnDiscountFactor' >value</param>
<param name='mnCMLineNo' >value</param>
<param name='szCMProgramID' >value</param>
<param name='szCMVersion' >value</param>
<param name='mnSupplierNo' >value</param>
<param name='szRelatedKitItemNo' >value</param>
<param name='mnKitMasterLineNo' >value</param>
<param name='mnComponentLineNo' >value</param>
<param name='mnRelatedKitComponent' >value</param>
<param name='mnNoOfCpntPerParent' >value</param>
```

<param name='cOverridePrice' >value</param>
<param name='cOverrideCost' >value</param>
<param name='szUserID' >value</param>
<param name='jdDateUpdated' >value</param>
<param name='mnWKOrderTotal' >value</param>
<param name='mnWKForeignOrderTotal' >value</param>
<param name='mnWKTotalCost' >value</param>
<param name='mnWKForeignTotalCost' >value</param>
<param name='cWKProcessingType' >value</param>
<param name='cWKSourceOfData' >value</param>
<param name='cWKCheckAvailability' >value</param>
<param name='mnLastLineNoAssigned' >value</param>
<param name='cStockingType' >value</param>
<param name='szOriginalOrderKeyCo' >value</param>
<param name='szOriginalOrderNo' >value</param>
<param name='szOriginalOrderType' >value</param>
<param name='mnOriginalOrderLineNo' >value</param>
<param name='cParentItmMethdOfPriceCalcn' >value</param>
<param name='szLandedCost' >value</param>
<param name='mnWKSuppressProcess' >value</param>
<param name='mnShortItemNo' >value</param>
<param name='mnWKRelatedOrderProcess' >value</param>
<param name='mnSODLineNo' >value</param>
<param name='mnPriceAdjRevLevel' >value</param>
<param name='szSalesOrderFlags' >value</param>
<param name='mnSODDocNo' >value</param>
<param name='szSODDocType' >value</param>
<param name='szSODOrderCo' >value</param>
<param name='szTransferOrderToBranch' >value</param>
<param name='mnDomesticDetachedAdj' >value</param>
<param name='mnForeignDetachedAdj' >value</param>

```
<param name='mnSODWFLineNo' >value</param>
<param name='szGeneric2CharString' >value</param>
<param name='mnTOEPOExchangeRate' >value</param>
<param name='szTOEPOCurrencyCode' >value</param>
<param name='mnDRPKeyId' >value</param>
<param name='mnSoldToCust' >value</param>
<param name='szF4201BranchPlant' >value</param>
<param name='szSoldToCurrencyCode' >value</param>
<param name='cConsolidationFlag' >value</param>
<param name='jdPriceEffectiveDate' >value</param>
<param name='mnWOWFLineNo' >value</param>
<param name='mnLineNoIncrement' >value</param>
<param name='mnParentWFLineNo' >value</param>
<param name='cStatusInWarehouse' >value</param>
<param name='cBypassCommitments' >value</param>
<param name='szProductSource' >value</param>
<param name='szProductSourceType' >value</param>
<param name='mnSequenceNumber' >value</param>
<param name='szAgreementNumber' >value</param>
<param name='mnAgreementSupplement' >value</param>
<param name='mnAgreementsFound' >value</param>
<param name='szModeOfTransport' >value</param>
<param name='szDutyStatus' >value</param>
<param name='szLineofBusiness' >value</param>
<param name='jdPromisedShip' >value</param>
<param name='szEndUse' >value</param>
<param name='mnTOEPOExchangeRate' >value</param>
<param name='szPriceCode1' >value</param>
<param name='szPriceCode2' >value</param>
<param name='szPriceCode3' >value</param>
<param name='szItemFlashMessage' >value</param>
```

```
<param name='szCompanyKeyRelated' >value</param>
<param name='szRelatedPoSoNumber >value</param>
<param name='szRelatedOrderType >value</param>
<param name='mnRelatedPoSoLineNo>value</param>
<param name='cGenericChar3 >value</param>
<param name='mnProfitMargin >value</param>
<param name='mnQuantityAvailable >value</param>
<param name='cRequestScheduleFlag >value</param>
<param name='cOrderProcessType >value</param>
<param name='cGenericChar2 >value</param>
<param name='mnSODRelatedJobNumber >value</param>
<param name='szGenericString >value</param>
<param name='mnCarrier >value</param>
<param name='szGenericString2_DL02 >value</param>
<param name='mnGenericMathNumeric1_MATH01 >value</param>
<param name='mnGenericMathNumeric2_MATH02 >value</param>
<param name='mnItemVolume_ITVL >value</param>
<param name='szVolumeUOM_VLUM >value</param>
<param name='szRevenueBusinessUnit >value</param>
<param name='szCustomerPO_VR01 >value</param>
<param name='szReference2Vendor_VR02 >value</param>
<param name='mnProcessID >value</param>
<param name='mnTransactionID >value</param>
</params >
<onError abort='no'>\
  </onError >
</callMethod >
<callMethod name=' F4211FSEndDoc'app='NetCommerce' runOnError='no'>
<params>
<param name='mnCMJobNo ` idref='j1`></param>
<param name='mnSalesOrderNo' >value</param>
```

```
<param name='szCMComputerID' idref='c2' >< /param>
<param name='cCMErrorCondition' >value</param>
<param name='szOrderType' >value</param>
<param name='szKeyCompany' >value</param>
<param name='mnOrderTotal' >value</param>
<param name='mnForeignOrderTotal' >value</param>
<param name='szBaseCurrencyCode' >value</param>
<param name='szProgramID' >value</param>
<param name='szWorkstationID' >value</param>
<param name='szCMProgramID' >value</param>
<param name='szCMVersion' >value</param>
<param name='mnTimeOfDay' >value</param>
<param name='mnTotalCost' >value</param>
<param name='mnForeignTotalCost' >value</param>
<param name='cSuppressRlvBlnktFlag' >value</param>
<param name='cWKSkipProcOptions' >value</param>
<param name='mnWKRelatedOrderProcess' >value</param>
<param name='cCMUseWorkFiles' >value</param>
<param name='mnEDIDocNo' >value</param>
<param name='szEDIKeyCo' >value</param>
<param name='szEDIDocType' >value</param>
<param name='cCMProcessEdits' >value</param>
<param name='cGenericChar2' >value</param>
<param name='mnSODRelatedJobNumber' >value</param>
<param name='cGenericChar1_EV01' >value</param>
<param name='mnGenericMathNumeric2_MATH02' >value</param>
<param name='szGenericString1_DL01' >value</param>
<param name='szGenericString2_DL02' >value</param>
<param name='mnProcessID' >value</param>
<param name='mnTransactionID' >value</param>
<params/>
```

```
<onError abort='no'>\
<callMethod name=' F4211ClearWorkFile'app='NetCommerce'
runOnError='yes'>
<params>
  <param name='mnJobNo` idref='j1`></param>
<param name='szComputerID' idref='c2' >< /param>
 <param name='mnFromLineNo' >value</param>
<param name='mnThruLineNo' >value</param>
<param name='cClearHeaderWF' >value</param>
<param name='cClearDetailWF' >value</param>
<param name='szProgramID' >value</param>
<param name='mnWKRelatedOrderProcess' >value</param>
<param name='szCMVersion' >value</param>
<param name='cGenericChar1_EV01' >value</param>
<param name='szGenericString1_DL01' >value</param>
     <param name='mnSODRelatedJobNumber' >value</param>
<param name='mnProcessID' >value</param>
<param name='mnTransactionID' >value</param>
</params>
</callMethod>
</onError>
</callMethod>
<returnParams version='value' messagetype='messsage name"failure Destination='queuename'
successDestination='queuename>
<param name='long description' idref='value'/param>
</returnParams>
<onError>
<callMethod name=' F4211ClearWorkFile'app='NetCommerce'
runOnError='yes'>
<params>
  <param name='mnJobNo` idref='j1`></param>
```

```
<param name='szComputerID' idref='c2' >< /param>
 <param name='mnFromLineNo' >value</param>
<param name='mnThruLineNo' >value</param>
<param name='cClearHeaderWF' >value</param>
<param name='cClearDetailWF' >value</param>
<param name='szProgramID' >value</param>
<param name='mnWKRelatedOrderProcess' >value</param>
<param name='szCMVersion' >value</param>
<param name='cGenericChar1_EV01' >value</param>
<param name='szGenericString1_DL01' >value</param>
    <param name='mnSODRelatedJobNumber' >value</param>
<param name='mnProcessID' >value</param>
<param name='mnTransactionID' >value</param>
</params>
</callMethod>
</onError>
</jdeRequest>
```

例:送信顧客作成 XML 要求および応答フォーマット(すべてのパラメータ)

次の例に、要求と応答の両方について、送信 XML フォーマットとすべてのパラメータを示します。

```
□This format will return all columns for the customer master (F0101Z2)□
<?xml version='1.0' ?>
  <jdeRequest type='trans' user='user' pwd='password' environment='environment' session=' ` sessio
nidle='300'>
 <transaction action='transactionInfo' type='JDEAB'>
 <key>
  <column name='EdiUserId'>value</column>
  <column name='EdiBatchNumber'>value</column>
  <column name='EdiTransactNumber'>value</column>
        </key>
 </transaction>
</jdeRequest>
```

```
<?xml version='1.0' encoding='utf-8' ?>
<jdeResponse type='trans' user='user' session='session'   environment='env'>
        <transaction type='JDEAB' action='transactionInfo'>
                <returnCode code='0'>XML Request OK</returnCode>
        <key>
    <column name='EdiUserId'></column>
      <column name='EdiBatchNumber'></column>
     </key>
     <table name='F0101Z2' type='detail'>
        <column name='EdiUserId'></column>
      <column name='EdiBatchNumber'></column>
      <column name='EdiTransactNumber'></column>
      <column name='EdiLineNumber'></column>
      <column name='EdiDocumentType'></column>
   <column name='TypeTransaction'></column>
   <column name='EdiTranslationFormat'></column>
  <column name='EdiTransmissionDate'></column>
  <column name='DirectionIndicator'></column>
  <column name='EdiDetailLinesProcess'></column>
  <column name='EdiSuccessfullyProcess'></column>
  <column name='TradingPartnerId'></column>
  <column name='TransactionAction'></column>
  <column name='AddressNumber'></column>
  <column name='AlternateAddressKey'></column>
  <column name='TaxId'></column>
  <column name='NameAlpha'></column>
  <column name='DescripCompressed'></column>
  <column name='CostCenter'></column>
  <column name='StandardIndustryCode'></column>
  <column name='LanguagePreference'>< /column>
  <column name='AddressType1'></column>
  <column name='CreditMessage'></column>
  <column name='PersonCorporationCode'></column>
  <column name='AddressType2'></column>
  <column name='AddressType3'></column>
  <column name='AddressType4'></column>
  <column name='AddressTypeReceivables'></column>
  <column name='AddressType5'></column>
  <column name='AddressTypePayables'></column>
  <column name='AddTypeCode4Purch'></column>
```

```
  <column name='MiscCode3'></column>
  <column name='AddressTypeEmployee'></column>
  <column name='SubledgerInactiveCode'></column>
  <column name='DateBeginningEffective'></column>
  <column name='AddressNumber1st'></column>
  <column name='AddressNumber2nd'></column>
  <column name='AddressNumber3rd'></column>
  <column name='AddressNumber4th'></column>
  <column name='AddressNumber6th'></column>
  <column name='AddressNumber5th'></column>
  <column name='ReportCodeAddBook001'></column>
  <column name='ReportCodeAddBook002'></column>
  <column name='ReportCodeAddBook003'></column>
  <column name='ReportCodeAddBook004'></column>
  <column name='ReportCodeAddBook005'></column>
  <column name='ReportCodeAddBook006'></column>
  <column name='ReportCodeAddBook007'></column>
  <column name='ReportCodeAddBook008'></column>
  <column name='ReportCodeAddBook009'></column>
  <column name='ReportCodeAddBook010'></column>
  <column name='ReportCodeAddBook011'></column>
<column name='ReportCodeAddBook012'></column>
<column name='ReportCodeAddBook013'></column>
<column name='ReportCodeAddBook014'></column>
<column name='ReportCodeAddBook015'></column>
<column name='ReportCodeAddBook016'></column>
<column name='ReportCodeAddBook017'></column>
<column name='ReportCodeAddBook018'></column>
<column name='ReportCodeAddBook019'></column>
<column name='ReportCodeAddBook020'></column>
<column name='CategoryCodeAddressBook2'></column>
<column name='CategoryCodeAddressBk22'></column>
<column name='CategoryCodeAddressBk23'></column>
<column name='CategoryCodeAddressBk24'></column>
<column name='CategoryCodeAddressBk25'></column>
<column name='CategoryCodeAddressBk26'></column>
<column name='CategoryCodeAddressBk27'></column>
<column name='CategoryCodeAddressBk28'></column>
<column name='CategoryCodeAddressBk29'></column>
<column name='CategoryCodeAddressBk30'></column>
```

```
<column name='GlBankAccount'></column>
<column name='TimeScheduledIn'></column>
<column name='DateScheduledIn'></column>
<column name='ActionMessageControl'></column>
<column name='NameRemark'></column>
<column name='CertificateTaxExempt'></column>
<column name='TaxId2'></column>
<column name='Kanjialpha'></column>
<column name='UserReservedCode'></column>
<column name='UserReservedDate'></column>
<column name='UserReservedAmount'></column>
<column name='UserReservedNumber'></column>
<column name='UserReservedReference'></column>
<column name='NameMailing'></column>
<column name='SecondaryMailingName'></column>
<column name='AddressLine1'></column>
<column name='AddressLine2'></column>
<column name='AddressLine3'></column>
<column name='AddressLine4'></column>
<column name='ZipCodePostal'></column>
<column name='City'></column>
<column name='Country'></column>
<column name='State'></column>
<column name='CountyAddress'></column>
<column name='PhoneAreaCode1'></column>
<column name='PhoneNumber'></column>
<column name='PhoneNumberTyp1'></column>
<column name='PhoneAreaCode2'></column>
<column name='PhoneNumber1'></column>
<column name='PhoneNumberTyp2'></column>
<column name='TransactionOriginator'></column>
<column name='UserId'></column>
<column name='ProgramId'></column>
<column name='WorkStationId'></column>
<column name='DateUpdated'></column>
<column name='TimeOfDay'></column>
<column name='TimeLastUpdated'></column>
      </table>
         </transaction>
</jdeResponse>
```

XMLフォーマット例（デフォルト値）

次の例は、特定のフォーマットに使用します。

例：受信受注オーダーXML フォーマット

この例では、ERP デフォルト値を使用します。外部エンティティが指定していないパラメータは省略されます。

```
<?xml version="1.0" encoding="utf-8" ?>
- <jdeRequest type="callmethod" user="JDE" pwd="JDE" environment="PRD733">
- <callMethod name="GetLocalComputerId" app="NetComm" runOnError="no">
- <params>
  <param name="szMachineKey" id="m2" />
  </params>
  <onError abort="yes" />
  </callMethod>
- <callMethod name="F4211FSBeginDoc" app="NetComm" runOnError="no">
- <params>
  <param name="mnCMJobNumber" id="j1" />
  <param name="cCMDocAction">A</param>
  <param name="cCMProcessEdits">1</param>
  <param name="szCMComputerID" idref="c2" />
  <param name="cCMUpdateWriteToWF">2</param>
  <param name="szCMProgramID">NetComm</param>
  <param name="szCMVersion">ZJDE0001</param>
  <param name="szOrderType">SO</param>
  <param name="szBusinessUnit">M30</param>
  <param name="mnAddressNumber">4242</param>
  <param name="jdOrderDate">2000/01/21</param>
  <param name="szReference">10261</param>
  <param name="cApplyFreightYN">Y</param>
  <param name="szCurrencyCode">CAD</param>
  <param name="cWKSourceOfData" />
  <param name="cWKProcMode" />
  <param name="mnWKSuppressProcess">0</param>
  </params>
- <onError abort="yes">
```

```
- <callMethod name="F4211ClearWorkFile" app="NetComm" runOnEr
ror="yes">
- <params>
  <param name="mnJobNo" idref="j1" />
  <param name="szComputerID" idref="c2" />
  <param name="mnFromLineNo">0</param>
  <param name="mnThruLineNo">0</param>
  <param name="cClearHeaderWF">2</param>
  <param name="cClearDetailWF">2</param>
  <param name="szProgramID">NetComm</param>
  <param name="szCMVersion">ZJDE0001</param>
  </params>
  </callMethod>
  </onError>
  </callMethod>
- <callMethod name="F4211FSEditLine" app="NetComm" runOnError="yes">
- <params>
  <param name="mnCMJobNo" idref="j1" />
  <param name="cCMLineAction">A</param>
  <param name="cCMProcessEdits">1</param>
  <param name="cCMWriteToWFFlag">2</param>
  <param name="szCMComputerID" idref="c2" />
  <param name="szItemNo">1001</param>
  <param name="mnQtyOrdered">1</param>
  <param name="cSalesTaxableYN">N</param>
  <param name="szTransactionUOM">EA</param>
  <param name="szCMProgramID">NetComm</param>
  <param name="szCMVersion">ZJDE0001</param>
  <param name="cWKSourceOfData" />
  </params>
  <onError abort="no" />
  </callMethod>
- <callMethod name="F4211FSEditLine" app="NetComm" runOnError="yes">
- <params>
  <param name="mnCMJobNo" idref="j1" />
  <param name="cCMLineAction">A</param>
  <param name="cCMProcessEdits">1</param>
  <param name="cCMWriteToWFFlag">2</param>
  <param name="szCMComputerID" idref="c2" />
  <param name="szItemNo">1001</param>
  <param name="mnQtyOrdered">10</param>
  <param name="cSalesTaxableYN">N</param>
  <param name="szTransactionUOM">EA</param>
  <param name="szCMProgramID">NetComm</param>
  <param name="szCMVersion">ZJDE0001</param>
  <param name="cWKSourceOfData" />
  </params>
  <onError abort="no" />
```

```
 </callMethod>
- <callMethod name="F4211FSEndDoc" app="NetComm" runOnError="no">
- <params>
  <param name="mnCMJobNo" idref="j1" />
  <param name="szCMComputerID" idref="c2" />
  <param name="szCMProgramID">NetComm</param>
  <param name="szCMVersion">ZJDE0001</param>
  <param name="cCMUseWorkFiles">2</param>
  </params>
- <onError abort="no">
- <callMethod name="F4211ClearWorkFile" app="NetComm" runOnEr
ror="yes">
- <params>
  <param name="mnJobNo" idref="j1" />
  <param name="szComputerID" idref="c2" />
  <param name="mnFromLineNo">0</param>
  <param name="mnThruLineNo">0</param>
  <param name="cClearHeaderWF">2</param>
  <param name="cClearDetailWF">2</param>
  <param name="szProgramID">NetComm</param>
  <param name="szCMVersion">ZJDE0001</param>
  </params>
  </callMethod>
  </onError>
  </callMethod>
  <returnParams failureDestination="ERROR.Q" successDestination="SUCCESS.Q" ru
nOnError="yes" />
- <onError abort="yes">
- <callMethod name="F4211ClearWorkFile" app="NetComm" runOnError="yes">
- <params>
  <param name="mnJobNo" idref="j1" />
  <param name="szComputerID" idref="c2" />
  <param name="mnFromLineNo">0</param>
  <param name="mnThruLineNo">0</param>
  <param name="cClearHeaderWF">2</param>
  <param name="cClearDetailWF">2</param>
  <param name="szProgramID">NetComm</param>
  <param name="szCMVersion">ZJDE0001</param>
  </params>
  </callMethod>
  </onError>
  </jdeRequest>
```

XML フォーマット例（Z イベント）

次の例に、Z イベントとリアルタイム・イベントのサンプル XML を示します。

例:Z イベント XML フォーマット

次の例は、Z ファイル・イベント XML ドキュメントを示します。

```
<?xml version='1.0' encoding='utf-8' ?>
<jdeResponse type='trans' user='JDE' environment='XDEVNIS2'>
  <transaction type='JDESC' action='transactionInfo'>
    <returnCode code='0'>XML Request OK</returnCode>
    <key>
      <EdiUserId>KW6803955</EdiUserId>
      <EdiBatchNumber>16319</EdiBatchNumber>
      <EdiTransactNumber>106053</EdiTransactNumber>
    </key>
    <F4201Z1 type='header'>
      <EdiUserId>KW6803955</EdiUserId>
      <EdiBatchNumber>16319</EdiBatchNumber>
      <EdiTransactNumber>106053</EdiTransactNumber>
      <EdiLineNumber>1.000</EdiLineNumber>
      <EdiDocumentType>SO</EdiDocumentType>
      <TypeTransaction>JDESC</TypeTransaction>
      <EdiTranslationFormat> </EdiTranslationFormat>
      <EdiTransmissionDate></EdiTransmissionDate>
      <DirectionIndicator>2</DirectionIndicator>
      <EdiDetailLinesProcess>0</EdiDetailLinesProcess>
      <EdiSuccessfullyProcess>Y</EdiSuccessfullyProcess>
      <TradingPartnerId> </TradingPartnerId>
      <TransactionAction>UA</TransactionAction>
      <CompanyKeyOrderNo>00200</CompanyKeyOrderNo>
      <DocumentOrderInvoiceE>6559</DocumentOrderInvoiceE>
      <OrderType>SO</OrderType>
      <OrderSuffix>000</OrderSuffix>
```

```
<CostCenter>        M30</CostCenter>
<Company>00200</Company>
<CompanyKeyOriginal> </CompanyKeyOriginal>
<OriginalPoSoNumber> </OriginalPoSoNumber>
<OriginalOrderType> </OriginalOrderType>
<CompanyKeyRelated> </CompanyKeyRelated>
<RelatedPoSoNumber> </RelatedPoSoNumber>
<RelatedOrderType> </RelatedOrderType>
<AddressNumber>4242</AddressNumber>
<AddressNumberShipTo>4242</AddressNumberShipTo>
<AddressNumberParent>4242</AddressNumberParent>
<DateRequestedJulian>2005/05/05</DateRequestedJulian>
<DateTransactionJulian>2005/05/05</DateTransactionJulian>
<PromisedDeliveryDate>2005/05/05</PromisedDeliveryDate>
<DateOriginalPromisde>2005/05/05</DateOriginalPromisde>
<ActualDeliveryDate></ActualDeliveryDate>
<CancelDate></CancelDate>
<DatePriceEffectiveDate>2005/05/05</DatePriceEffectiveDate>
<DatePromisedPickJu>2005/05/05</DatePromisedPickJu>
<DatePromisedShipJu></DatePromisedShipJu>
<Reference1> </Reference1>
<Reference2Vendor> </Reference2Vendor>
<DeliveryInstructLine1> </DeliveryInstructLine1>
<DeliveryInstructLine2> </DeliveryInstructLine2>
<PrintMessage1> </PrintMessage1>
<PaymentTermsCode01> </PaymentTermsCode01>
<PaymentInstrumentA> </PaymentInstrumentA>
```

```
<PriceAdjustmentScheduleN> </PriceAdjustmentScheduleN>
<PricingGroup>PREFER</PricingGroup>
<DiscountTrade>.000</DiscountTrade>
<PercentRetainage1>.000</PercentRetainage1>
<TaxArea1>DEN</TaxArea1>
<TaxExplanationCode1>S</TaxExplanationCode1>
<CertificateTaxExempt> </CertificateTaxExempt>
<AssociatedText> </AssociatedText>
<PriorityProcessing>0</PriorityProcessing>
<BackordersAllowedYN>Y</BackordersAllowedYN>
<SubstitutesAllowedYN>Y</SubstitutesAllowedYN>
<HoldOrdersCode> </HoldOrdersCode>
<PricePickListYN>Y</PricePickListYN>
<InvoiceCopies>0</InvoiceCopies>
<NatureOfTransaction> </NatureOfTransaction>
<BuyerNumber>0</BuyerNumber>
<Carrier>0</Carrier>
<ModeOfTransport> </ModeOfTransport>
<ConditionsOfTransport> </ConditionsOfTransport>
<RouteCode> </RouteCode>
<StopCode> </StopCode>
<ZoneNumber> </ZoneNumber>
<ContainerID> </ContainerID>
<FreightHandlingCode> </FreightHandlingCode>
<ApplyFreightYN>Y</ApplyFreightYN>
<ApplyFreight> </ApplyFreight>
<FreightCalculatedYN> </FreightCalculatedYN>
```

<MergeOrdersYN> </MergeOrdersYN>
<CommissionCode1>6001</CommissionCode1>
<RateCommission1>5.000</RateCommission1>
<CommissionCode2>0</CommissionCode2>
<RateCommission2>.000</RateCommission2>
<ReasonCode> </ReasonCode>
<PostQuantities> </PostQuantities>
<AmountOrderGross>134.97</AmountOrderGross>
<AmountTotalCost>.00</AmountTotalCost>
<UnitOfMeasureWhtDisp> </UnitOfMeasureWhtDisp>
<UnitOfMeasureVolDisp> </UnitOfMeasureVolDisp>
<AuthorizationNoCredit> </AuthorizationNoCredit>
<AcctNoCrBank> </AcctNoCrBank>
<DateExpired></DateExpired>
<SubledgerInactiveCode> </SubledgerInactiveCode>
<CorrespondenceMethod> </CorrespondenceMethod>
<CurrencyMode>F</CurrencyMode>
<CurrencyCodeFrom>BEF</CurrencyCodeFrom>
<CurrencyConverRateOv>33.8180588</CurrencyConverRateOv>
<LanguagePreference>E</LanguagePreference>
<AmountForeignOpen>4564.42</AmountForeignOpen>
<AmountForeignTotalC>.00</AmountForeignTotalC>
<OrderedBy> </OrderedBy>
<OrderTakenBy> </OrderTakenBy>
<UserReservedCode> </UserReservedCode>
<UserReservedDate></UserReservedDate>
<UserReservedAmount>.00</UserReservedAmount>

```
    <UserReservedNumber>0</UserReservedNumber>
    <UserReservedReference> </UserReservedReference>
    <UserId>KW6803955</UserId>
    <ProgramId> </ProgramId>
    <WorkStationId>STI5</WorkStationId>
    <DateUpdated>2000/08/22</DateUpdated>
    <TimeOfDay>134435</TimeOfDay>
  </F4201Z1>
  <F4211Z1 type='detail'>
    <EdiUserId>KW6803955</EdiUserId>
    <EdiBatchNumber>16319</EdiBatchNumber>
    <EdiTransactNumber>106053</EdiTransactNumber>
    <EdiLineNumber>1.000</EdiLineNumber>
    <EdiDocumentType>SO</EdiDocumentType>
    <TypeTransaction>JDESC</TypeTransaction>
    <EdiTranslationFormat> </EdiTranslationFormat>
    <EdiTransmissionDate></EdiTransmissionDate>
    <DirectionIndicator>2</DirectionIndicator>
    <EdiDetailLinesProcess>0</EdiDetailLinesProcess>
    <EdiSuccessfullyProcess>N</EdiSuccessfullyProcess>
    <TradingPartnerId> </TradingPartnerId>
    <TransactionAction>UA</TransactionAction>
    <CompanyKeyOrderNo>00200</CompanyKeyOrderNo>
    <DocumentOrderInvoiceE>6559</DocumentOrderInvoiceE>
    <OrderType>SO</OrderType>
    <LineNumber>1.000</LineNumber>
    <OrderSuffix>000</OrderSuffix>
```

<CostCenter> M30</CostCenter>
<Company>00200</Company>
<CompanyKeyOriginal> </CompanyKeyOriginal>
<OriginalPoSoNumber> </OriginalPoSoNumber>
<OriginalOrderType> </OriginalOrderType>
<OriginalLineNumber>.000</OriginalLineNumber>
<CompanyKeyRelated> </CompanyKeyRelated>
<RelatedPoSoNumber> </RelatedPoSoNumber>
<RelatedOrderType> </RelatedOrderType>
<RelatedPoSoLineNo>.000</RelatedPoSoLineNo>
<ContractNumberDistributi> </ContractNumberDistributi>
<ContractSupplementDistri>0</ContractSupplementDistri>
<ContractBalancesUpdatedY> </ContractBalancesUpdatedY>
<AddressNumber>4242</AddressNumber>
<AddressNumberShipTo>4242</AddressNumberShipTo>
<AddressNumberParent>4242</AddressNumberParent>
<DateRequestedJulian>2005/05/05</DateRequestedJulian>
<DateTransactionJulian>2005/05/05</DateTransactionJulian>
<PromisedDeliveryDate>2005/05/05</PromisedDeliveryDate>
<DateOriginalPromisde>2005/05/05</DateOriginalPromisde>
<ActualDeliveryDate></ActualDeliveryDate>
<DateInvoiceJulian></DateInvoiceJulian>
<CancelDate></CancelDate>
<DtForGLAndVouch1></DtForGLAndVouch1>
<DateReleaseJulian>2005/05/05</DateReleaseJulian>
<DatePriceEffectiveDate>2005/05/05</DatePriceEffectiveDate>
<DatePromisedPickJu>2005/05/05</DatePromisedPickJu>

<DatePromisedShipJu></DatePromisedShipJu>
<Reference1> </Reference1>
<Reference2Vendor> </Reference2Vendor>
<IdentifierShortItem>60003</IdentifierShortItem>
<Identifier2ndItem>1001</Identifier2ndItem>
<Identifier3rdItem>1001</Identifier3rdItem>
<Location> </Location>
<Lot> </Lot>
<FromGrade> </FromGrade>
<ThruGrade> </ThruGrade>
<FromPotency>.000</FromPotency>
<ThruPotency>.000</ThruPotency>
<DaysPastExpiration>0</DaysPastExpiration>
<DescriptionLine1>Bike Rack - Trunk Mount</DescriptionLine1>
<DescriptionLine2> </DescriptionLine2>
<LineType>S</LineType>
<StatusCodeNext>540</StatusCodeNext>
<StatusCodeLast>520</StatusCodeLast>
<CostCenterHeader> M30</CostCenterHeader>
<ItemNumberRelatedKit> </ItemNumberRelatedKit>
<LineNumberKitMaster>.000</LineNumberKitMaster>
<ComponentNumber>.0</ComponentNumber>
<RelatedKitComponent>0</RelatedKitComponent>
<NumbOfCpntPerParent>0</NumbOfCpntPerParent>
<SalesReportingCode1> </SalesReportingCode1>
<SalesReportingCode2> </SalesReportingCode2>
<SalesReportingCode3> </SalesReportingCode3>

<SalesReportingCode4> </SalesReportingCode4>
<SalesReportingCode5> </SalesReportingCode5>
<PurchasingReportCode1> </PurchasingReportCode1>
<PurchasingReportCode2> </PurchasingReportCode2>
<PurchasingReportCode3> </PurchasingReportCode3>
<PurchasingReportCode4>240</PurchasingReportCode4>
<PurchasingReportCode5> </PurchasingReportCode5>
<UnitOfMeasureAsInput>EA</UnitOfMeasureAsInput>
<UnitsTransactionQty>3</UnitsTransactionQty>
<UnitsQuantityShipped>3</UnitsQuantityShipped>
<UnitsQuanBackorHeld>0</UnitsQuanBackorHeld>
<UnitsQuantityCanceled>0</UnitsQuantityCanceled>
<UnitsQuantityFuture>0</UnitsQuantityFuture>
<UnitsOpenQuantity>0</UnitsOpenQuantity>
<QuantityShippedToDate>0</QuantityShippedToDate>
<QuantityRelieved>0</QuantityRelieved>
<CommittedHS>S</CommittedHS>
<OtherQuantity12> </OtherQuantity12>
<AmtPricePerUnit2>44.9900</AmtPricePerUnit2>
<AmountExtendedPrice>134.97</AmountExtendedPrice>
<AmountOpen1>.00</AmountOpen1>
<PriceOverrideCode> </PriceOverrideCode>
<TemporaryPriceYN> </TemporaryPriceYN>
<UnitOfMeasureEntUP>EA</UnitOfMeasureEntUP>
<AmtListPricePerUnit>44.9900</AmtListPricePerUnit>
<AmountUnitCost>32.1000</AmountUnitCost>
<AmountExtendedCost>96.30</AmountExtendedCost>

<CostOverrideCode> </CostOverrideCode>
<ExtendedCostTransfer>.0000</ExtendedCostTransfer>
<PrintMessage1> </PrintMessage1>
<PaymentTermsCode01> </PaymentTermsCode01>
<PaymentInstrumentA> </PaymentInstrumentA>
<BasedonDate> </BasedonDate>
<DiscountTrade>.000</DiscountTrade>
<TradeDiscountOld>.0000</TradeDiscountOld>
<PriceAdjustmentScheduleN> </PriceAdjustmentScheduleN>
<PricingCategory> </PricingCategory>
<PricingCategoryLevel1> </PricingCategoryLevel1>
<DiscountFactor>1.0000</DiscountFactor>
<DiscountFactorTypeOr> </DiscountFactorTypeOr>
<DiscntApplicationType> </DiscntApplicationType>
<DiscountCash>.000</DiscountCash>
<CompanyKey> </CompanyKey>
<DocVoucherInvoiceE>0</DocVoucherInvoiceE>
<DocumentType> </DocumentType>
<OriginalDocumentNo>0</OriginalDocumentNo>
<OriginalDocumentType> </OriginalDocumentType>
<DocumentCompanyOriginal> </DocumentCompanyOriginal>
<PickSlipNumber>0</PickSlipNumber>
<DeliveryNumber>0</DeliveryNumber>
<PromationNumber> </PromationNumber>
<DraftNumber>0</DraftNumber>
<TaxableYN>N</TaxableYN>
<TaxArea1>DEN</TaxArea1>

```
<TaxExplanationCode1>S</TaxExplanationCode1>
<AssociatedText> </AssociatedText>
<PriorityProcessing>0</PriorityProcessing>
<ResolutionCodeBC> </ResolutionCodeBC>
<BackordersAllowedYN>Y</BackordersAllowedYN>
<SubstitutesAllowedYN>Y</SubstitutesAllowedYN>
<PartialShipmntsAllowY>Y</PartialShipmntsAllowY>
<LineofBusiness> </LineofBusiness>
<EndUse> </EndUse>
<DutyStatus> </DutyStatus>
<CommodityCode> </CommodityCode>
<NatureOfTransaction> </NatureOfTransaction>
<PrimaryLastVendorNo>4343</PrimaryLastVendorNo>
<BuyerNumber>8444</BuyerNumber>
<Carrier>0</Carrier>
<ModeOfTransport> </ModeOfTransport>
<ConditionsOfTransport> </ConditionsOfTransport>
<RouteCode> </RouteCode>
<StopCode> </StopCode>
<ZoneNumber> </ZoneNumber>
<ContainerID> </ContainerID>
<FreightHandlingCode> </FreightHandlingCode>
<ApplyFreightYN>Y</ApplyFreightYN>
<ApplyFreight> </ApplyFreight>
<FreightCalculatedYN> </FreightCalculatedYN>
<RateCodeFrieghtMisc> </RateCodeFrieghtMisc>
<RateTypeFreightMisc> </RateTypeFreightMisc>
```

<ShippingCommodityClass> </ShippingCommodityClass>
<ShippingConditionsCode> </ShippingConditionsCode>
<SerialNumberLot> </SerialNumberLot>
<UnitOfMeasurePrimary>EA</UnitOfMeasurePrimary>
<UnitsPrimaryQtyOrder>3</UnitsPrimaryQtyOrder>
<UnitOfMeasureSecondary>EA</UnitOfMeasureSecondary>
<UnitsSecondaryQtyOr>3</UnitsSecondaryQtyOr>
<UnitOfMeasurePricing>EA</UnitOfMeasurePricing>
<AmountUnitWeight>240.0000</AmountUnitWeight>
<WeightUnitOfMeasure>OZ</WeightUnitOfMeasure>
<AmountUnitVolume>6.7500</AmountUnitVolume>
<VolumeUnitOfMeasure>FC</VolumeUnitOfMeasure>
<RepriceBasketPriceCat> </RepriceBasketPriceCat>
<OrderRepriceCategory> </OrderRepriceCategory>
<OrderRepricedIndicator> </OrderRepricedIndicator>
<InventoryCostingMeth>07</InventoryCostingMeth>
<AllocateByLot> </AllocateByLot>
<GlClass>IN30</GlClass>
<Century>20</Century>
<FiscalYear1>5</FiscalYear1>
<LineStatus> </LineStatus>
<SalesOrderStatus01> </SalesOrderStatus01>
<SalesOrderStatus02> </SalesOrderStatus02>
<SalesOrderStatus03> </SalesOrderStatus03>
<SalesOrderStatus04> </SalesOrderStatus04>
<SalesOrderStatus05> </SalesOrderStatus05>
<SalesOrderStatus06> </SalesOrderStatus06>

```
<SalesOrderStatus07> </SalesOrderStatus07>
<SalesOrderStatus08> </SalesOrderStatus08>
<SalesOrderStatus09> </SalesOrderStatus09>
<SalesOrderStatus10> </SalesOrderStatus10>
<SalesOrderStatus11> </SalesOrderStatus11>
<SalesOrderStatus12> </SalesOrderStatus12>
<SalesOrderStatus13> </SalesOrderStatus13>
<SalesOrderStatus14> </SalesOrderStatus14>
<SalesOrderStatus15> </SalesOrderStatus15>
<Salesperson1>6001</Salesperson1>
<SalespersonCommission1>5.000</SalespersonCommission1>
<Salesperson2>0</Salesperson2>
<SalespersonCommission2>.000</SalespersonCommission2>
<ApplyCommissionYN>Y</ApplyCommissionYN>
<CommissionCategory> </CommissionCategory>
<ReasonCode> </ReasonCode>
<GrossWeight>.0000</GrossWeight>
<UnitOfMeasureGrossWt> </UnitOfMeasureGrossWt>
<AcctNoInputMode> </AcctNoInputMode>
<AccountId> </AccountId>
<PurchasingCostCenter> </PurchasingCostCenter>
<ObjectAccount> </ObjectAccount>
<Subsidiary> </Subsidiary>
<LedgerType> </LedgerType>
<Subledger> </Subledger>
<SubledgerType> </SubledgerType>
<CodeLocationTaxStat> </CodeLocationTaxStat>
```

```
    <PriceCode1> </PriceCode1>
    <PriceCode2> </PriceCode2>
    <PriceCode3> </PriceCode3>
    <StatusInWarehouse> </StatusInWarehouse>
    <WoOrderFreezeCode> </WoOrderFreezeCode>
    <CorrespondenceMethod> </CorrespondenceMethod>
    <CurrencyCodeFrom>BEF</CurrencyCodeFrom>
    <CurrencyConverRateOv>33.8180588</CurrencyConverRateOv>
    <AmountListPriceForeign>1521.4745</AmountListPriceForeign>
    <AmtForPricePerUnit>1521.4745</AmtForPricePerUnit>
    <AmountForeignExtPrice>4564.42</AmountForeignExtPrice>
    <AmountForeignUnitCost>1085.5597</AmountForeignUnitCost>
    <AmountForeignExtCost>3256.68</AmountForeignExtCost>
    <UserReservedCode> </UserReservedCode>
    <UserReservedDate></UserReservedDate>
    <UserReservedAmount>.00</UserReservedAmount>
    <UserReservedNumber>0</UserReservedNumber>
    <UserReservedReference> </UserReservedReference>
    <TransactionOriginator>KW6803955</TransactionOriginator>
    <UserId>KW6803955</UserId>
    <ProgramId>XMLtest</ProgramId>
    <WorkStationId>STI5</WorkStationId>
    <DateUpdated>2000/08/22</DateUpdated>
    <TimeOfDay>134435</TimeOfDay>
  </F4211Z1>
  <F49211Z1 type='additionalHeader'>
    <EdiUserId>KW6803955</EdiUserId>
```

<EdiBatchNumber>16319</EdiBatchNumber>
<EdiTransactNumber>106053</EdiTransactNumber>
<EdiLineNumber>1.000</EdiLineNumber>
<EdiDocumentType>SO</EdiDocumentType>
<TypeTransaction>JDESC</TypeTransaction>
<EdiTranslationFormat> </EdiTranslationFormat>
<EdiTransmissionDate></EdiTransmissionDate>
<DirectionIndicator>2</DirectionIndicator>
<EdiDetailLinesProcess>0</EdiDetailLinesProcess>
<EdiSuccessfullyProcess>N</EdiSuccessfullyProcess>
<TradingPartnerId> </TradingPartnerId>
<TransactionAction>UA</TransactionAction>
<DocumentOrderInvoiceE>6559</DocumentOrderInvoiceE>
<OrderType>SO</OrderType>
<CompanyKeyOrderNo>00200</CompanyKeyOrderNo>
<LineNumber>1.000</LineNumber>
<CostCenterTrip> </CostCenterTrip>
<TripNumber>0</TripNumber>
<DateLoaded></DateLoaded>
<DispatchGrp> </DispatchGrp>
<BulkPackedFlag>P</BulkPackedFlag>
<Distance>0</Distance>
<UnitOfMeasure> </UnitOfMeasure>
<DeferredEntriesFlag> </DeferredEntriesFlag>
<AmountDeferredCost>.0000</AmountDeferredCost>
<AmountForeignDeferredCos>.0000</AmountForeignDeferredCos>
<AmountDeferredRevenue>.0000</AmountDeferredRevenue>

<AmountForeignDeferredRe>.0000</AmountForeignDeferredRe>
<AaiTableNumber>0</AaiTableNumber>
<ScheduledInvoiceDate></ScheduledInvoiceDate>
<InvoiceCycleCode> </InvoiceCycleCode>
<LoadConfirmDate></LoadConfirmDate>
<TimeLoad>0</TimeLoad>
<DeliveryConfirmDate></DeliveryConfirmDate>
<UnitsPrimaryCommittedQua>0</UnitsPrimaryCommittedQua>
<UnitofMeasureCommittedQu> </UnitofMeasureCommittedQu>
<Temperature>.00</Temperature>
<StrappingTemperatureUnit> </StrappingTemperatureUnit>
<Density>.00</Density>
<DensityTypeAtStandardTem> </DensityTypeAtStandardTem>
<DensityTemperature>.00</DensityTemperature>
<DensityTemperatureUnit> </DensityTemperatureUnit>
<VolumeCorrectionFactors>.0000</VolumeCorrectionFactors>
<PriceatAmbiantorStandard>A</PriceatAmbiantorStandard>
<PricingBasedOnDate> </PricingBasedOnDate>
<UnitsInvoiceQuantity>0</UnitsInvoiceQuantity>
<StockTotalinPrimaryUOM>0</StockTotalinPrimaryUOM>
<UnitofMeasure6> </UnitofMeasure6>
<AmbientResult>0</AmbientResult>
<UnitofMeasure3> </UnitofMeasure3>
<WeightResult>0</WeightResult>
<UnitofMeasure5> </UnitofMeasure5>
<VendorFreightCalculatedY> </VendorFreightCalculatedY>
<CustomerFreightCalculate> </CustomerFreightCalculate>

<AmountCustomerFreightCha>.0000</AmountCustomerFreightCha>
<AmountVendorFreightCharg>.0000</AmountVendorFreightCharg>
<PrimaryVehicleId> </PrimaryVehicleId>
<RegistrationLicenseNumbe> </RegistrationLicenseNumbe>
<CostCenterArDefault> </CostCenterArDefault>
<FlightNumber> </FlightNumber>
<Destination> </Destination>
<AircraftType> </AircraftType>
<Origin> </Origin>
<TimeElapsed>0</TimeElapsed>
<ShipmentNumberB73>0</ShipmentNumberB73>
<AddressNumberIssued>6074</AddressNumberIssued>
<PaymentTermsCode01> </PaymentTermsCode01>
<DocVoucherInvoiceE>0</DocVoucherInvoiceE>
<DocumentType> </DocumentType>
<CompanyKey> </CompanyKey>
<CurrencyConverRateOv>-1.0000000</CurrencyConverRateOv>
<CurrencyCodeFrom> </CurrencyCodeFrom>
<TaxArea1>DEN</TaxArea1>
<TaxExplanationCode1> </TaxExplanationCode1>
<ForeignDomesticFlag> </ForeignDomesticFlag>
<FuelingPort> </FuelingPort>
<RegistrationIdentificati> </RegistrationIdentificati>
<DeliveryLocationN> </DeliveryLocationN>
<AuthorizationName> </AuthorizationName>
<NameAlpha> </NameAlpha>
<MeterTicket1> </MeterTicket1>

```
<UnitsBeginningThroughput>0</UnitsBeginningThroughput>
<ClosingReading1>0</ClosingReading1>
<MeterTicket2> </MeterTicket2>
<UnitsBeginningThroughpu2>0</UnitsBeginningThroughpu2>
<ClosingReading2>0</ClosingReading2>
<MeterTicket3> </MeterTicket3>
<UnitsBeginningThroughpu3>0</UnitsBeginningThroughpu3>
<ClosingReading3>0</ClosingReading3>
<DateArrival></DateArrival>
<TimeArrival>0</TimeArrival>
<DateDeparture></DateDeparture>
<TimeDeparture>0</TimeDeparture>
<DateStartJobJulian></DateStartJobJulian>
<TimeBeginningHHMM>0</TimeBeginningHHMM>
<DateEnding></DateEnding>
<TimeStopHHMM>0</TimeStopHHMM>
<FutureUse01t> </FutureUse01t>
<FutureUse02t> </FutureUse02t>
<FutureUse03t> </FutureUse03t>
<FutureUse04> </FutureUse04>
<FutureUse05> </FutureUse05>
<FutureUseCode> </FutureUseCode>
<FutureUseQuantityt>0</FutureUseQuantityt>
<FutureUseDate></FutureUseDate>
<FutureUseUnitofMeasure> </FutureUseUnitofMeasure>
<UserReservedCode> </UserReservedCode>
<UserReservedDate></UserReservedDate>
```

```
      <UserReservedAmount>.00</UserReservedAmount>
      <UserReservedNumber>0</UserReservedNumber>
      <UserReservedReference> </UserReservedReference>
      <TransactionOriginator> </TransactionOriginator>
      <UserId>KW6803955</UserId>
      <ProgramId>XMLtest</ProgramId>
      <WorkStationId>STI5</WorkStationId>
      <DateUpdated>2000/08/22</DateUpdated>
      <TimeOfDay>134435</TimeOfDay>
    </F49211Z1>
  </transaction>
</jdeResponse>
```

例:リアルタイム・イベント・テンプレート

ここでは、リアルタイム・イベント・テンプレートの例を示します。この例のテンプレートは、アプリケーションで使用される正確なイベントとは対応していない場合があります。イベントには、サンプル・テンプレートにない値が含まれる可能性があります。

```
//The event must be described in the jdeResponse type element. The attribute type is always
"realTimeEvent". The attributes for user and environment always correspond to the username and
environment that generated the event.

<?xml version="1.0" encoding="utf-8" ?>
<jdeResponse type="realTimeEvent" user="" session="28980548.1019684006"
    environment="">
<event>
<header>

//Code for the header information follows. <eventVersion> is always 1.0, <type> corresponds to the event
type, <application> corresponds to the application that created the event, and <version> to the version of the
application. The <sessionID> is unique for every event. The <scope> is the value of the argument scope that
was sent to the real-time event API during creation of the event. The <codepage> element is for encoding of
the elements; in the sample utf-8 is used. The remaining header elements are self-explanatory.

 <eventVersion>1.0</eventVersion>
 <type>RTSOOUT</type>
 <user />
 <application />
 <version />
 <sessionID />
 <environment />
 <host />
 <sequenceID />
 <date />
 <time />
 <scope />
 <codepage>utf-8</codepage>
```

</header>

//The body contains details that describe one data structure for each element. The body contains the date of creation, the name of the file that is creating the data structure, time of creation, and the DSTMPL name of the ERP data structure. Type is type of partial event (added as an argument to jdeIEO–_EventAdd), executionOrder increases in the real generated event from 1 to elementCount, and parameterCount is the number of fields in the data structure. In the following example code, there are three data structures: D34A1050C, D4202150C, and D4202150B. Each data structure is followed by detail elements. When you create an event, the element value is the value of the field, for example: <szNameAlpha type="String">ABC</szNameAlpha >

<body elementCount="3">
<detail date="" name="" time="" type="" DSTMPL="D34A1050C"
    executionOrder="" parameterCount="25">
<szNameAlpha type="String" />
<mnParentAddressNumber type="Double" />
<szSecondItemNumber type="String" />
<szThirdItemNumber type="String" />
<cPriorityProcessing type="Character" />
<cBackOrdersAllowed type="Character" />
<cOrderShippedFlag type="Character" />
<cTransferDirectShipFlag type="Character" />
<cCommitted type="Character" />
<mnDaysBeforeExpiration type="Double" />
<szPurchaseCategoryCode1 type="String" />
<szPurchaseCategoryCode2 type="String" />
<szPurchaseCategoryCode3 type="String" />
<szPurchaseCategoryCode4 type="String" />
<szRelatedOrderNumber type="String" />
<szRelatedOrderType type="String" />
<szRelatedOrderKeyCompany type="String" />
<szPlanningUnitOfMeasure type="String" />
<mnPlanningQuantity type="Double" />
<cAPSFlag type="Character" />
<cAPSSupplyDemandFlag type="Character" />
<jdDateUpdated type="Date" />
<mnTimeUpdated type="Double" />
<szShipComplete type="String" />
<mnRelatedOrderLineNumber type="Double" />
</detail>
<detail date="" name="" time="" type="" DSTMPL="D4202150C"
    executionOrder="" parameterCount="94">
<cOrderAction type="Character" />
<szOrderType type="String" />
<szOrderCompany type="String" />
<mnLineNumber type="Double" />
<szDetailBranchPlant type="String" />
<mnShipToAddressNumber type="Double" />
<jdTransactionDate type="Date" />
<jdRequestedDate type="Date" />
<jdScheduledPickDate type="Date" />
<jdPromisedShipDate type="Date" />
<jdPromisedDeliveryDate type="Date" />
<jdCancelDate type="Date" />
<jdPriceEffectiveDate type="Date" />

```
<mnQuantityOrdered type="Double" />
<mnQuantityShipped type="Double" />
<mnQuantityBackOrdered type="Double" />
<mnQuantityCanceled type="Double" />
<szTransactionUnitOfMeasure type="String" />
<mnUnitPrice type="Double" />
<mnExtendedPrice type="Double" />
<mnForeignUnitPrice type="Double" />
<mnForeignExtPrice type="Double" />
<cPriceOverrideCode type="Character" />
<cTaxableYN type="Character" />
<szPriceAdjustmentSchedule type="String" />
<mnDiscountPercentage type="Double" />
<szPaymentTerms type="String" />
<cPaymentInstrument type="Character" />
<szCurrencyCode type="String" />
<szItemNumber type="String" />
<mnShortItemNumber type="Double" />
<szDescriptionLine1 type="String" />
<szDescriptionLine2 type="String" />
<szLineType type="String" />
<szLastStatus type="String" />
<szNextStatus type="String" />
<szLocation type="String" />
<szLot type="String" />
<szLineofBusiness type="String" />
<szEndUse type="String" />
<szDutyStatus type="String" />
<szPrintMessage1 type="String" />
<szFreightHandlingCode type="String" />
<mnItemWeight type="Double" />
<szWeightUnitOfMeasure type="String" />
<szModeOfTransport type="String" />
<mnCarrier type="Double" />
<szSubledger type="String" />
<cSubledgerType type="Character" />
<szPriceCode1 type="String" />
<szPriceCode2 type="String" />
<szPriceCode3 type="String" />
<szSalesReportingCode1 type="String" />
<szSalesReportingCode2 type="String" />
<szSalesReportingCode3 type="String" />
<szSalesReportingCode4 type="String" />
<szSalesReportingCode5 type="String" />
<szOriginalPoSoNumber type="String" />
<szOriginalOrderType type="String" />
<szOriginalOrderCompany type="String" />
<mnOriginalOrderLineNumber type="Double" />
<jdDateUpdated type="Date" />
<mnTimeOfDay type="Double" />
<mnPickSlipNumber type="Double" />
<mnInvoiceDocNumber type="Double" />
<szInvoiceDocType type="String" />
<szInvoceDocCompany type="String" />
<szUserReservedCode type="String" />
```

<jdUserReservedDate type="Date" />
<mnUserReservedNumber type="Double" />
<mnUserReservedAmount type="Double" />
<szUserReservedReference type="String" />
<mnUnitCost type="Double" />
<mnExtendedCost type="Double" />
<mnForeignUnitCost type="Double" />
<mnForeignExtCost type="Double" />
<mnOrderNumber type="Double" />
<szSupplierReference type="String" />
<jdOriginalPromisdDate type="Date" />
<mnAdjustmentRevisionLevel type="Double" />
<mnLastIndex type="Double" />
<szRelatedPoSoNumber type="String" />
<szRelatedOrderType type="String" />
<szRelatedOrderCompany type="String" />
<mnRelatedPoSoLineNo type="Double" />
<szPricingUnitOfMeasure type="String" />
<szTaxArea type="String" />
<szTaxExplanationCode type="String" />
<szPartnerItemNo type="String" />
<szCatalogItem type="String" />
<szUPCNumber type="String" />
<szShipToDescriptive type="String" />
<szSoldToDescriptive type="String" />
<szProductItem type="String" />
</detail>
<detail date="" name="" time="" type="" DSTMPL="D4202150B"
executionOrder="" parameterCount="66">
<cOrderAction type="Character" />
<mnOrderNumber type="Double" />
<szOrderType type="String" />
<szOrderCompany type="String" />
<szHeaderBranchPlant type="String" />
<szCompany type="String" />
<szOriginalPoSoNumber type="String" />
<szOrderedBy type="String" />
<szOrderTakenBy type="String" />
<mnSoldToAddressNumber type="Double" />
<szSoldToNameMailing type="String" />
<szSoldToAddressLine1 type="String" />
<szSoldToAddressLine2 type="String" />
<szSoldToAddressLine3 type="String" />
<szSoldToAddressLine4 type="String" />
<szSoldToZipCode type="String" />
<szSoldToCity type="String" />
<szSoldToCounty type="String" />
<szSoldToState type="String" />
<szSoldToCountry type="String" />
<mnShipToAddressNumber type="Double" />
<szShipToNameMailing type="String" />
<szShipToAddressLine1 type="String" />
<szShipToAddressLine2 type="String" />
<szShipToAddressLine3 type="String" />
<szShipToAddressLine4 type="String" />

```
<szShipToZipCode type="String" />
<szShipToCity type="String" />
<szShipToCounty type="String" />
<szShipToState type="String" />
<szShipToCountry type="String" />
<jdTransactionDate type="Date" />
<jdRequestedDate type="Date" />
<jdCancelDate type="Date" />
<szReference type="String" />
<szDeliveryInstructLine1 type="String" />
<szDeliveryInstructLine2 type="String" />
<szPrintMessage type="String" />
<szFreightHandlingCode type="String" />
<mnCommissionCode1 type="Double" />
<mnCommissionCode2 type="Double" />
<mnRateCommission1 type="Double" />
<mnRateCommission2 type="Double" />
<mnDiscountTrade type="Double" />
<szPaymentTerms type="String" />
<cPaymentInstrument type="Character" />
<szCurrencyCode type="String" />
<mnCurrencyConverRate type="Double" />
<szTaxArea type="String" />
<szTaxExplanationCode type="String" />
<mnOrderTotal type="Double" />
<mnForeignOrderTotal type="Double" />
<szUserReservedCode type="String" />
<jdUserReservedDate type="Date" />
<mnUserReservedAmount type="Double" />
<mnUserReservedNumber type="Double" />
<szUserReservedReference type="String" />
<szHoldCode type="String" />
<cQuoteFlag type="Character" />
<jdScheduledPickDate type="Date" />
<jdPromisedShipDate type="Date" />
<jdOriginalPromisdDate type="Date" />
<cCurrencyMode type="Character" />
<szShipToDescriptive type="String" />
<szSoldToDescriptive type="String" />
<cPublishToXPIxFlag type="Character" />
</detail>
</body>
</event>
</jdeResponse>
```

次の表に、ERP のタイプとイベントのマッピングを示します。

CHAR	Character（文字）
STRING	String（文字列）
MATH_numeric	Double
JDEDATE	Dat″
SHORT	Int
INT	Int
USHORT	Int
LONG	Long
ULONG	Long
ID	″Long
ID2	Long
BOOL	BOOL（ブール）

用語解説

AAI:「自動仕訳」を参照。

アクション・メッセージ(action message):ERP では、ユーザーは ERP のフォーム、アプリケーション、および該当するデータへのショートカットが含まれたメッセージ(システム生成またはユーザー生成)を受け取ることができる。たとえば、総勘定元帳転記によってユーザーにアクション・エラー・メッセージが送られると、そのユーザーはエラーがある仕訳(または入力)にメッセージから直接アクセスできる。これはワークフロー戦略の中核であり、ERP からのアクション・メッセージの場合と、サードパーティの電子メール・システムからのアクション・メッセージの場合がある。

アクティベータ(activator):Solution Explorer において、ディレクタによって自動化される、順序どおりに整列された子タスクを持つ親タスク。

ActiveX:OLE に基づいたコンピューティング・テクノロジー。Web ブラウザやその他のアプリケーションの Java アプレット形式の機能を使用可能にする(Java は、現時点では Web ブラウザに限定されている)。ActiveX において Java アプレットに相当するのは ActiveX コントロールである。これらのコントロールは、コントロールを「包含」するプログラムに対して計算、通信、およびデータ処理の機能をもたらす。たとえば、特定の Web ブラウザ、Microsoft Office プログラム、および Visual Basic または Visual C++によって開発されたあらゆる機能がある。

拡張(advance):オブジェクト管理ワークベンチ内のプロジェクトの状況変更。プロジェクトを拡張すると、この状況変更によって、あるサーバーから別のサーバーにオブジェクトを移動したり、プロジェクト・オブジェクトのチェックアウトを防ぐなどのアクションや条件がトリガーされることがある。

英数字(alphanumeric character):英字、数字、記号の組合せであり、データを表すために使用される。「数字」および「特殊文字」の反対語。

API:「アプリケーション・プログラミング・インターフェイス」を参照。

APPL:「アプリケーション」を参照。

アプレット(applet):ユーティリティ・プログラムや限られた機能を持つスプレッドシートなどの小型アプリケーション。通常は、プログラミング言語である Java に関連付けられており、このコンテキストでは、ワークステーション上の Web ブラウザから値を渡すことができるインターネット対応アプリケーションを指す。

アプリケーション(application):コンピュータ業界では、「実行可能ファイル」の同義語。ERP における対話型またはバッチ・アプリケーションは、メニューから起動して買掛管理や受注オーダー処理などのビジネス・タスクを実行できる 1 組の関連フォームに関するプログラミングを含む DLL。システムとも呼ぶ。

アプリケーション開発者(application developer):ERP ツールセットを使用して ERP アプリケーションを開発するプログラマ。

アプリケーション・プログラミング・インターフェイス(API:application programming interface):あるプログラムの機能にアクセスするために、他のプログラムから実行できるソフトウェア関数呼出し。

アプリケーション・ワークスペース(application workspace):アプリケーション上ですべての関連フォームが表示される領域。

監査証跡(audit trail):処理済みのトランザクションに関して検証可能な詳細な履歴。履歴は、元の文書、トランザクション入力、レコード転記からなり、通常は最後にレポートが生成される。

自動仕訳（AAI：automatic accounting instruction）勘定科目表の勘定科目を参照するコード。AAI では、売掛管理、買掛管理、財務レポート、一般会計のような各システム間のインターフェイスなど、仕訳を自動生成するプログラムに関するルールを定義する。一般会計システムとインターフェイスする各システムには、AAI がある。たとえば、AAI では、〈General Ledger Post〉プログラムに対して、特定の費用勘定科目の借方と、特定の買掛金勘定科目の貸方に転記するように指示できる。

バッチ見出し(batch header)：トランザクションまたはレコードのバッチを識別し、制御するための情報。

バッチ・ジョブ(batch job)：レポートの印刷やファイルの除去など、処理のために投入すると、処理中は 1 単位として扱われるタスクまたはタスクのグループ。コンピュータ・システムでは、バッチ・ジョブは自動的に、またはわずかなユーザー操作によって実行される。

バッチ処理(batch processing)：システムがジョブ待ち行列からジョブを選択し、処理して、出力を出力待ち行列に送る方法。「対話型処理」の反対語。

バッチ・サーバー(batch server)：クライアント、アプリケーション・サーバー、またはエンタープライズ・サーバーの代わりに、ERP のバッチ処理要求（UBE とも呼ぶ）が実行されるサーバー。通常、バッチ・サーバーにはデータベースは格納されておらず、対話型アプリケーションも実行されない。

バッチ・タイプ(batch type)：関連トランザクションが関連付けられているシステムを示し、処理用に選択されるレコードを制御するために、バッチ・ジョブに割り当てられるコード。たとえば、〈Post General Journal〉プログラムでは、バッチ・タイプが 0 の未転記トランザクション・バッチのみが転記用に選択される。

即時バッチ・オブ・ワン(batch-of-one immediate)：処理をクライアント・ワークステーション上で実行してから、後続の処理のためにすべてを一度にサーバー・アプリケーションに投入するトランザクション方式。バッチ処理はサーバー上で実行されるため、クライアント・アプリケーションは引き続き他のタスクを実行できる。「ダイレクト接続」、「オフライン処理」も参照。

BDA：「Business View Design Aid（ビジネス・ビュー設計支援）」を参照。

バイナリ文字列（BSTR：binary string）：OLE オートメーション・データ操作関数によって使用されるプレフィックス付きの文字列。バイナリ文字列は、32 ビット Windows プラットフォーム上ではダブルバイト(Unicode)のワイド文字列である。

ブール論理オペランド(Boolean Logic Operand)：J.D. Edwards レポート・プログラムでは、[Relationship] フィールドのパラメータ。このブール論理オペランドは、特定のレコードまたはパラメータを比較するようにシステムに対して指示する。有効なオプションは次のとおり。

EQ	同等(=)
LT	より小さい(<)
LE	以下(<=)
GT	より大きい(>)
GE	以上(>=)
NE	同等(=)ではない
NL	より小さく(<)はない
NG	より大きく(>)はない

ブラウザ(browser)：World Wide Web が送信した情報を翻訳するクライアント・アプリケーション。クライアントが World Wide Web 情報を受信し、デスクトップ上で操作し、表示するには、ブラウザを使用する必要がある。Web ブラウザとも呼ぶ。

BSFN：「ビジネス関数」を参照。

BSTR：「バイナリ文字列」を参照。

BSVW：「ビジネス・ビュー」を参照。

ビジネス関数(business function)：通常は、複数のアプリケーションで再利用可能なビジネス・ルールやロジックがカプセル化された集合。ビジネス関数では、トランザクションやそのサブセット（在庫チェック、作業オーダーの発行など）を実行することができる。また、ビジネス関数には API も含まれているため、フォーム、データベース・トリガー、または ERP 以外のアプリケーションからも呼び出すことができる。ビジネス関数は、他のビジネス関数のみでなく、アプリケーションを構成するフォーム、イベント・ルール、および他の構成要素と組み合わせることができる。ビジネス関数の作成には、イベント・ルール、または C などの第三世代言語を使用する。ビジネス関数の例としては、Credit Check（与信チェック）や Item Availability（在庫照会）がある。

イベント・ルール・ビジネス関数(business function event rule)：「イベント・ルール・ビジネス関数」を参照。

ビジネス・ビュー(business view)：データベース・テーブルのデータにアクセスするために ERP アプリケーションで使用されるビュー。ビジネス・ビューは、アプリケーションやレポートに使用されるデータが入っている 1 つまたは複数のテーブルから特定のカラムを選択する手段である。ビジネス・ビューでは、特定のローは選択されず、物理データは含まれない。あくまでもデータ処理の手段として使用されるビューである。

ビジネス・ビュー設計支援（BDA：Business View Design Aid）：ビジネス・ビューを作成、修正、コピー、および印刷する ERP GUI ツール。このツールではグラフィカル・ユーザー・インターフェイスが使用される。

カテゴリ・コード(category code)：ユーザー定義コードでの、未定義カテゴリの一時的なタイトル。たとえば、異なる販売地域の指定コードを追加する場合、カテゴリ・コード 4 を Sales Region（販売地域）に変更し、有効なコードとして E(East)、W(West)、N(North)、および S(South)を定義できる。レポート・コードと呼ぶこともある。

セントラル・オブジェクト(central object)：セントラル・ロケーションに常駐するオブジェクト。セントラル・オブジェクト・データ・ソースとセントラル C コンポーネントの 2 つの部分から構成される。セントラル・オブジェクト・データ・ソースには、リレーショナル・データベースに保管される ERP のスペックが格納される。セントラル C コンポーネントはビジネス関数のソース、ヘッダー、オブジェクト、ライブラリ、および DLL ファイルを含み、通常はデプロイメント・サーバー上のディレクトリに保管される。この 2 つの部分がセントラル・オブジェクトを構成する。

チェックイン・ロケーション(check-in location)：パッケージとそのレプリケート・オブジェクト・セットが格納されるディレクトリ構造上の位置。通常は、¥¥deploymentserver¥release¥path_code¥package¥packagename である。このパスの下のサブディレクトリには、ビジネス関数用のセントラル C コンポーネント（ソース、インクルード、オブジェクト、ライブラリ、および DLL ファイル）が格納される。

子(child)：「親/子フォーム」を参照。

クライアント/サーバー(client/server)：別々のマシンで実行されるプロセス間の関係。サーバー・プロセスはソフトウェアの提供側であり、クライアントは提供されるサービスの消費側である。本質的に、クライアント/サーバーでは、サービスという概念に基づいて機能が明確に分離される。サーバーは、同時に多数のクライアントにサービスを提供し、共有リソースへのアクセスを調整することができる。クライアントとサーバーの間には、それぞれ多数対 1 の関係が存在する。クライアントは、常にサービスを要求してダイアログを実行する。サーバーは、クライアントからの要求を受動的に待機する。

CNC：「コンフィギュラブル・ネットワーク・コンピューティング」を参照。

コンポーネント(component):Portal では、ワークスペース内に表示されるカプセル化されたオブジェクト。ポータル・コンポーネント。

コンフィギュラブル・クライアント・エンジン(configurable client engine):ユーザーにインターフェイス・レベルでの柔軟性をもたらす。ユーザーは、カラムを手軽に移動し、さまざまなデータ・ビューのタブを設定し、ニーズに合わせてグリッドのサイズを設定することができる。また、Windows 95 ベースや Windows NT ベースのインターフェイスのみでなく、Web ブラウザも組込可能になる。

コンフィギュラブル・ネットワーク・コンピューティング(CNC:configurable network computing):単一コード・ベースで構成される対話型アプリケーションとバッチ・アプリケーションを、複数のサーバー・プラットフォームと SQL データベースからなる TCP/IP ネットワーク上で実行できるようにするアプリケーション・アーキテクチャ。アプリケーションは、再利用可能なビジネス関数と、それに関連してネットワーク上で動的に構成可能なデータで構成される。ビジネス全体の目標は、組織構造、業務、および相互に依存しないテクノロジーを変更できるように、将来を見込んだ環境を提供することである。

固定情報(constant):ユーザーが設定し、そのシステムで関連プログラムによる情報処理を標準化するために使用されるパラメータやコード。固定情報の例としては、部品表のオンライン確認や、固定労務間接費の原価への組込みなどがある。

コントロール(control):ユーザーがアプリケーションと対話できるようにするデータ入力ポイント。たとえば、チェックボックス、プルダウン・リスト、ハイパーボタン、入力フィールドなどの機能は、コントロールである。

コア製品(core):J.D. Edwards ソフトウェアのセントラル・システムおよびファンデーション・システム。一般会計、買掛管理、売掛管理、住所録、財務レポート、財務モデルと配布、およびバック・オフィスなどがある。

CRP:カンファレンス・ルーム・パイロット。

カスタム・グリッドライン(custom gridlines):合計など、データベースからは取り込まれないグリッド・ロー。合計をグリッドに表示するには、値を集計し、合計を表示するためのカスタム・グリッドラインを挿入する。そのためには、システム関数 Insert Grid Row Buffer を使用する。

データ辞書(data dictionary):データ項目の定義とスペックを保管および管理するための方法。
J.D. Edwards はアクティブなデータ辞書、つまり実行時にアクセスされる辞書を持っている。

データ・マート(data mart):部署レベルの意思決定支援データベース。通常は、組織から連結され、調整されたデータ・ソースとして機能するエンタープライズ・データ・ウェアハウスからデータが表示される。データ・マートは、リレーショナル・データベースでもマルチディメンション・データベースでもかまわない。

データ・レプリケーション(data replication):レプリケート環境では、データの複数コピーが複数のマシン上で保守される。データを「所有」する単一ソースが必要である。これにより、基本ロケーションにデータの最新コピーが適用されてから、必要に応じてレプリケートされることが保証される。これに対して、データの単純コピーの場合、コピーはセントラル・ロケーションから保守されるのではなく、ソースとの依存関係なしに存在する。

データ・ソース(data source):コンピュータ上で実行されるデータベース管理システムの特定のインスタンス。データ・ソースは、オブジェクト構成マネージャ(OCM)とオブジェクト・マップ(OM)を通じて管理される。

データ構造体(data structure):オブジェクト間で情報を渡すときに使用されるデータ項目のグループ。たとえば、2 つのフォーム間、フォームとビジネス関数の間、またはレポートとビジネス関数の間でやりとるすることができる。

データ・ウェアハウス(data warehouse):部署レベルの意思決定支援のためのクエリーおよびレポート用に、データ・マートに分散される前に、複数データベースのデータを調整、連結するために使用されるデータベース。通常は、運用データベースとデータ・マートの間に位置する専用サーバーに常駐する大型リレーショナル・データベースである。

データ・ウェアハウジング(data warehousing):データ・ウェアハウジングには、基本的に、操作データ・ソースをアンロードして、意思決定支援(レポートとクエリー)のために排他的に使用されるターゲット・データベースにロードする処理が含まれる。意思決定支援環境には、二重データベース、拡張分析データベース、エンタープライズ・データ・ウェアハウスなど、さまざまなものがある。

データベース(database):システムで使用、保管されるすべての情報の集合。絶えず更新される。データベースにより、オンラインによる情報の作成、保管、インデックス作成、および相互参照が可能になる。

データベース・ドライバ(database driver):アプリケーションを特定のデータベース管理システムに接続するソフトウェア。

データベース・サーバー(database server):データが保管されるサーバー。データベース・サーバーには、ERP ロジックは存在しない。

DCE:「分散コンピューティング環境」を参照。

DD:「データ辞書」を参照。

デフォルト(default):何も指定されていない場合に使用されるコード、数値、またはパラメータ値。

明細(detail):レコードまたはトランザクションを構成する特定の情報およびデータ。「集計」の反対語。

グリッド(detail area):ERP アプリケーションおよび機能に備えられた、スプレッドシート・グリッドに類似するコントロール。データの複数のローを一度に表示、追加、または更新する。

ダイレクト接続(direct connect):クライアント・アプリケーションとサーバー・アプリケーションが対話形式で情報を直接やりとりするトランザクション方式。「即時バッチ・オブ・ワン」、「オフライン処理」も参照。

ディレクタ(director):ユーザーがタスクを完了するまでのプロセスをガイドする対話型ユーティリティ。

分散コンピューティング環境(DCE:distributed computing environment):複数のコンピュータ上で実行されるソフトウェアを、エンドユーザーに透過的な方法でシームレスに実行できるようにする統合ソフトウェア・サービス・セット。DCE は、ネットワーク上で実行されるコンピュータ間にセキュリティ、ディレクトリ、時間、リモート・プロシージャ・コール、およびファイルを提供する。

DLL:「ダイナミック・リンク・ライブラリ」を参照。

DS:「データ構造体」を参照。

DSTR:「データ構造体」を参照。

複製データベース(duplicated database):処理データの単純なオプションが入っている意思決定支援データベース。処理環境とレポート環境のパフォーマンス改善という長所がある。「拡張分析データベース」、「エンタープライズ・データ・ウェアハウス」も参照。

ダイナミック・リンク・ライブラリ(DLL:dynamic link library):実行可能ファイルの実行時に、実行可能ファイルにリンクしなくても、実行可能ファイルから呼び出されるように設計されたプログラム・モジュールのセット。通常は、共通して使用される関数が含まれている。

動的パーティショニング(dynamic partitioning):ロジックやデータをクライアント/サーバー・アーキテクチャ内で複数の階層に動的に分散する機能。

埋込みイベント・ルール(embedded event rule):特定のテーブルやアプリケーションに特有のイベント・ルール。たとえば、フォーム間の呼出し、処理オプションの値に基づくフィールドの非表示化、ビジネス関数の呼出しなどが含まれる。「イベント・ルール・ビジネス関数」の反対語。「イベント・ルール」も参照。

従業員ワーク・センター(employee work center):発信側のアプリケーションやユーザーに関係なく、すべての ERP メッセージ(システム生成とユーザー生成)を送受信するための中心。各ユーザーは、アクティブ・メッセージなど、ワークフローや他のメッセージを含むメールボックスを持つ。ワークフローの場合、メッセージ・センターは MAPI 準拠であり、ドラッグ&ドロップによる作業の割当て変更、エスカレーション、転送と返信、およびワークフローのモニタリングをサポートする。メッセージ・センターからのすべてのメッセージは、ERP メッセージまたは Microsoft Exchange を通じて表示できる。

カプセル化(encapsulation):オブジェクト内のデータへのアクセスと操作を、そのオブジェクトの定義に影響するプロシージャに限定する機能。

拡張分析データベース(enhanced analysis database):処理データのサブセットが入っているデータベース。拡張分析データベース内のデータには、レポート生成時間とクエリー応答時間を短縮するための簡単な計算および集計機能が含まれる。このソリューションは、ソース・データに外部データの追加を必要とする場合、または、動向分析や定期レポートの作成に履歴データを必要とする場合に適している。「複製データベース」、「エンタープライズ・データ・ウェアハウス」も参照。

エンタープライズ・データ・ウェアハウス(enterprise data warehouse):企業内のさまざまな分野からのデータを伴う複雑なソリューション。この環境には、調整、連結された企業内データのセントラル・リポジトリとして機能する大型リレーショナル・データベース(データ・ウェアハウス)が必要である。データ・マートには、このリポジトリからデータを取り込んで、部署レベルの意思決定を与える。「複製データベース」、「拡張分析データベース」も参照。

エンタープライズ・サーバー(enterprise server):データベース・サーバーとロジック・サーバー。「データベース・サーバー」を参照。ホストと呼ぶこともある。

ER:「イベント・ルール」を参照。

ERP:総合的でミッション・クリティカルなビジネス・アプリケーションと、そのアプリケーションを固有のビジネス要件およびテクノロジー要件に合わせて構成するための埋込みツールセットで構成されるスイート。ERP には J.D. Edwards 独自のアプリケーション・アーキテクチャである CNC テクノロジーが使用されている。このアーキテクチャによって、クライアント/サーバー機能の構成能力、調整能力、および安定面での能力が強化されている。

ERP アプリケーション(ERP application):ERP のビジネス機能を実行する対話型プロセスまたはバッチ・プロセス。再利用可能なビジネス関数と、プラットフォームに依存しない関連データで構成され、TCP/IP ネットワーク間で動的に構成できる。

ERP オブジェクト(ERP object):アプリケーションのビルドに使用され、再利用可能な個々のコード。オブジェクト・タイプには、テーブル、フォーム、ビジネス関数、データ辞書項目、バッチ処理、ビジネス・ビュー、イベント・ルール、バージョン、データ構造体、メディア・オブジェクトなどがある。「オブジェクト」も参照。

ERP プロセス(ERP process):ERP のクライアントとサーバーで処理要求を処理してトランザクションを実行できるようにする。クライアントは 1 つの処理を実行し、サーバーは複数のインスタンスを持つことができる。ERP プロセスを特定のタスク(ワークフロー・メッセージやデータ・レプリケーションなど)専用にできるため、特にサーバーがビジー状態の場合、重要なプロセスは待機する必要がなくなる。

ERP Web 開発コンピュータ(ERP Web development computer):次のコンポーネントが追加インストールされている標準 ERP Windows 開発者向けコンピュータ。

- JFC(0.5.1)
- Generator Package(Generator.Java および JDECOM.dll 付き)
- 解釈可能およびアプリケーション・コントロール/フォームを持つ R2

イベント(event):対話型アプリケーションやバッチ・アプリケーションの実行中に発生するアクション。たとえば、編集コントロールからのタブ移動、プッシュ・ボタンのクリック、フォームの初期化、またはの改ページの実行などである。GUI オペレーティング・システムでは、ミニプログラムを使用してフォーム内のユーザー・アクティビティを管理する。これらのミニプログラムにロジックを追加添付し、イベント・ルールを使用して、ERP アプリケーションやレポート内のイベント機能を強化することができる。

イベント・ルール(event rule):多くのプログラミング言語に伴う複雑な構文を使用しないで、複雑なビジネス・ロジックを作成するために使用する。これらのロジック・ステートメントをアプリケーションまたはデータベース・イベントにアタッチし、フォームへのアクセス、メニュー・バー・オプションの選択、レポートの改ページ、レコードの選択など、定義したイベントの発生時に実行させることができる。イベント・ルールでは、データを検査し、ユーザーにメッセージを送信し、ビジネス関数を呼び出すなど、多数のアクションを実行することができる。イベント・ルールは次の 2 つのタイプに分かれている。

埋込みイベント・ルール(Embedded event rule)

イベント・ルール・ビジネス関数(Named event rule)

実行可能ファイル(executable file):コンピュータのオペレーティング・システムから実行できるコンピュータ・プログラム。「アプリケーション」および「プログラム」の同義語。

エグジット(exit):1) 特定のキーまたは一連のキーを押すことにより、コンピュータ・プログラムを中断または終了すること。2) フォームに表示された、別のフォームにアクセスできるオプション・キーまたはファンクション・キー。

施設または機能(facility):1) 原価のトラッキング対象となる個々のビジネス・エンティティ。たとえば、倉庫所在地、ジョブ、プロジェクト、作業場、事業所などがある。ビジネスユニットと呼ぶこともある。2) Home Builder および ECS では、システム全体または統合された全システムに特殊機能を提供するコンピュータ言語ステートメントまたはプログラムの集合。たとえば、ドリームライターやファスターは機能である。

FDA:「フォーム設計ツール」を参照。

検索/表示(find/browse):このフォームの用途は次のとおり。

グリッド内の複数レコードの検索、表示、および選択。
レコードの削除。
別のフォームへのエグジット。
ほとんどのアプリケーションへのエントリ・ポイントを提供。

ファイヤウォール(firewall):企業がすべての着信メッセージをテストし、フィルタをかけ、ルーティングできるようにする一連のテクノロジー。ファイヤウォールは、企業内のセキュリティを維持するために使用される。

修正/検査(fix/inspect):既存のレコードを表示、追加、または修正するためのフォームのタイプ。このタイプのフォームには、グリッドは組み込まれていない。

フォーム(form):ERP の GUI(グラフィカル・ユーザー・インターフェイス)の 1 要素で、ユーザーがアプリケーションと対話するためのコントロールを持つ。フォームを使用して、ユーザーは情報を入力、選択、参照できる。ERP アプリケーションには複数のフォームを含むものもある。Microsoft Windows 用語では、フォームはダイアログ・ボックスと呼ばれる。

フォーム設計ツール(FDA:Form Design Aid)対話型のアプリケーションとフォームをビルドするための ERP GUI 開発ツール。

フォーム・インターコネクト(form interconnection):あるフォームから他のフォームにアクセスしてデータを渡すための機能。どんなイベントにも添付可能であるが、通常はボタンがクリックされたときに使用される。

フォーム・タイプ(form type):ERP で使用可能なフォーム・タイプは次のとおり。

検索/表示

修正/検査

見出し詳細

見出しなし詳細

メッセージ

親/子

検索/選択

第 4 世代言語(4GL:fourth generation language):必要なタスクに重点をおいて、その実行方法を決定するプログラミング言語。構造化照会言語(SQL)は、4GL の一例である。

グラフィカル・ユーザー・インターフェイス(GUI:graphical user interface):キャラクタ・ベースのインターフェイスに対する、グラフィック・ベースのコンピュータ・インターフェイス。キャラクタ・ベースのインターフェイスの例としては AS/400、GUI の例としては Microsoft Windows がある。グラフィカル・インターフェイスでは、ピクチャや他のグラフィック・イメージを使用して、コンピュータの操作方法に関するヒントを表示することができる。

グリッド(grid):「グリッド」を参照。

GUI:「グラフィカル・ユーザー・インターフェイス」を参照。

見出し(header):テーブルやフォームの先頭に表示される情報。この情報を使用して、後続のレコード・グループの制御情報が識別または提供される。

見出し/詳細(header/detail):2 つの異なるテーブルからのレコードを追加、変更、または削除するためのフォーム。通常、これらのテーブルは親/子関係を持つ。

見出しなし詳細(headerless detail):グリッドに表示された複数のレコードを処理するためのフォーム。グリッドにはデータを入力することができる。

隠し選択(hidden selections):メニューの[Selection]フィールドに"HS"と入力しないと表示されないメニュー項目。これらの項目は表示されないが、どのメニューからも使用できる。この種の項目には、[Display Submitted Jobs (33)]、[Display User Job Queue (42)]、[Display User Print Queue (43)]などがある。[Hidden Selections]ウィンドウには、ユーザー・ツール、オペレータ・ツール、プログラマ・ツールの 3 種類の項目が表示される。

ホスト(host):集中化されたコンピュータ・モデルでは、端末が情報をやりとりするときに依存する大規模タイムシェアリング・コンピュータ・システム。クライアント・サーバーでは、ユーザーは各自の処理の大部分は個々のコンピュータで作業し、ファイル管理、セキュリティ、およびプリンタ管理などのサービスはサーバーにアクセスすることで提供される。

HTML「ハイパーテキスト・マークアップ言語」を参照。

ハイパーテキスト・マークアップ言語(HTML:hypertext markup language):文書の物理的レイアウトではなく論理構造を指定するためのマークアップ言語。論理構造を指定すると、HTML ドキュメントはプラットフォームに依存しなくなる。HTML ドキュメントは、ブラウザのサポート機能を持つデスクトップであれば、どこででも表示することができる。HTML には、インターネット上やイントラネット・サイト上のどこかにある他の HTML ドキュメントへのアクティブ・リンクを組み込むことができる。

インデックス(index):値の順序とその一意性を表し、テーブルの各ローのデータに効率よくアクセスできるようにする。インデックスは、テーブルの 1 つまたは複数のカラムで構成される。

継承(inheritance):あるクラスが親クラスからデータやプロシージャ定義のすべてまたは一部を受け取る機能。継承によって、クラスとその関連コードが再利用可能になり、開発作業が強化される。

インストール・システム・コード(install system code):「システム・コード」を参照。

統合ツールセット(integrated toolset):既に総合的なビジネス・アプリケーションに埋め込まれている、業界で有力な ERP 独自のツールセット。このツールセットは、J.D. Edwards が ERP の対話型アプリケーションとバッチ・アプリケーションのビルドに使用しているものと同じである。ただし、この ERP 統合ツールセットでは、開発環境に比べると、はるかに多くのレポート作成機能や他のバッチ処理、変更管理、および基本的なデータ・ウェアハウジング機能が処理される。

対話型処理(interactive processing):システムに直接入力したコマンドに応答して発生する処理アクション。対話型処理中には、システムと直接通信しているため、要求の処理中に情報の追加入力を求められることがある。「オンライン」も参照。「バッチ処理」の反対語。

インターフェイス(interface):2 つ以上のコンピュータ・システム間のリンク。システムが相互に情報をやりとりできるようにする。

インターネット(Internet):電話回線や他のタイプのリモート・アクセスを通じてデスクトップ・クライアントがサーバー、アプリケーション、および情報にアクセスするための、世界中に分散されたネットワーク。

インタオペラビリティ(interoperability):さまざまなコンピュータ・システム、ネットワーク、オペレーティング・システム、およびアプリケーションを連動させて情報を共有する機能。

イントラネット(intranet):通常は、1 つの企業や組織に範囲が限定されたインターネットの小型版。イントラネットでは、インターネットの機能が使用され、設置場所は企業の裁量で決定される。

IP:無接続通信プロトコル。これ自体がデータグラム・サービスを提供する。データグラムとは、そのアドレス、およびルータに含まれるルーティング・テーブル情報に基づき、ルーターによって転送され

る自己完結型情報パケットである。TCP/IP ネットワーク上のどのノードにも、ネットワークと、ネットワーク上のローカル・ホストまたはノードを識別するアドレスが必要である。ほとんどの場合、ネットワーク管理者は、これらのアドレスを新規ワークステーションのインストール時に設定する。ただし、ワークステーションの場合は、動的に割り当てられるアドレスをブート時にサーバーにクエリーすることもできる。

IServer サービス(Iserver Service):J.D. Edwards が開発したサービス。このインターネット・サーバー・サービスは Web サーバーに常駐し、データベースからクライアントへの Java クラスのファイル配信を高速化するために使用される。

ISO 9000:ISO(国際標準化機構)により指定された一連の標準で、製品およびサービスの品質の指標として立案されたもの。

J.D. Edwards データベース(J.D. Edwards Database):「JDEBASE データベース・ミドルウェア」を参照。

Java:C のように、プラットフォーム間で高度な移植性を発揮するように設計されたインターネット実行可能言語。このプログラミング言語は、Sun Microsystems 社によって開発された。オペレーティング・システムやブラウザが Java 対応であれば、アプレット、つまり Java アプリケーションに Web ブラウザからアクセスでき、クライアントが使用できる(通常、Java は小規模な C++として記述される)。Java アプリケーションは、プラットフォームに依存しない。

Java データベース・コネクティビティ(JDBC:Java Database Connectivity):Sun Microsystems 社によって設定された、Java データベースへの標準的なアクセス方法。この標準により、任意の JDBC ドライバ・データベースを使用することができる。

JavaScript:Java に関連するスクリプト言語。ただし、Java とは異なり、JavaScript はオブジェクト指向言語ではないためコンパイルできない。

jde.ini:ERP の初期化に必要なランタイム設定を提供する J.D. Edwards ファイル(または AS/400 のメンバ)。ERP を実行する各マシンには、ファイル/メンバの特定バージョンを常駐させる必要がある。これには、ワークステーションとサーバーが含まれる。

JDEBASE データベース・ミドルウェア(JDEBASE Database Middleware):次の 2 つの主要な利点をもたらす J.D. Edwards 独自のデータベース・ミドルウェア・パッケージ。

> 複数データベースへのアクセス用のプラットフォーム独立型 API。これらの API には、次の 2 つの用途がある。
>
> - 対話型エンジンおよびバッチ・エンジンが、データ・ソース要求に応じてプラットフォーム特定の SQL を自動作成するために使用される。
> - C ビジネス関数の上級ライティングのためのオープン API として使用される。これらの API は、エンジンによって使用され、プラットフォーム固有の SQL が動的に生成される。
>
> クライアント/サーバー間、およびサーバー/サーバー間のデータベース・アクセス。そのために、ERP は Client Access 400、オープン・データベース・コネクティビティ(ODBC)など、さまざまなサードパーティ・データベース・ドライバと統合されている。

JDECallObject:ビジネス関数から他のビジネス関数を起動するためのアプリケーション・プログラミング・インターフェイス。

JDENET:J.D. Edwards 独自のミドルウェア・ソフトウェア。メッセージ処理ソフトウェア・パッケージである。

JDENet コミュニケーション・ミドルウェア(JDENET communications middleware):J.D. Edwards 独自の、ERP 向け通信ミドルウェア・パッケージ。ピア・ツー・ピア、メッセージ・ベース、ソケット・ベースのマルチ・プロセス通信用ミドルウェア・ソリューションである。ERP サポート・プラットフォームすべてのクライアントとサーバー間およびサーバー間コミュニケーションを処理する。

ジョブ待ち行列(job queue):バッチ処理を待機中のジョブのグループ。「バッチ処理」も参照。

ジャストインタイム・インストール(JITI:just in time installation):ERP でセントラル・オブジェクト・ロケーションからワークステーションにオブジェクトを動的にレプリケートする方法。

ジャストインタイム・レプリケーション(JITR:just in time replication):ERP でデータを個々のワークステーションにレプリケートする方法。ERP では、新規レコードはユーザーがそのデータを必要とする場合にのみレプリケート(挿入)される。変更、削除、および更新は、プル・レプリケーションを使用してレプリケートする必要がある。

キー(KEY):データベース・テーブル内の 1 つまたは複数のレコードを識別するカラムまたはその組合せ。

先行ゼロ(leading zero):J.D. Edwards システム内の特定の機能によって、入力した値の前に配置される一連のゼロ。通常、フィールドの指定の長さよりも小さい値を入力した場合に発生する。たとえば、8 桁の数値を入力できるフィールドに"4567"と入力すると、入力した 4 桁の数値の前に 4 つのゼロが挿入される。この結果、00004567 と表示される。

詳細レベル(level of detail):1) J.D. Edwards ソフトウェアのメニューの難易度区分。メニューの詳細レベルは次のとおり。

A 主要な製品のディレクトリ

B 製品グループ

1 基本操作

2 中級操作

3 上級操作

4 コンピュータ・オペレーション

5 プログラマ

6 メニュー・レベルの別名でも知られる上級プログラマ

2) 一般会計システムで勘定科目情報が集計されるレベル。最上位レベルは 1(最も詳細度が低い)、最下位レベルは 9(最も詳細度が高い)。

MAPI:「メッセージング・アプリケーション・プログラミング・インターフェイス」を参照。

マスター・テーブル(master table):システムの処理に必要な持続的データおよび情報を格納するためのデータベース・テーブル。マスター・テーブルには、支払済み税額、仕入先名、住所、従業員情報、ジョブ情報などのデータが含まれる。

メニュー(menu):採番された選択項目を表示するメニュー。各選択項目はプログラムまたはその他のメニューを表す。メニューから項目にアクセスするには、選択項目番号を入力してから[Enter]キーを押す。

メニュー・レベル(menu level):「詳細レベル」を参照。

メニュー・マスキング(menu masking):指定したメニューやメニュー項目に対する個々のユーザーのアクセスを制限できる J.D. Edwards システムの機密保護機能。権限を持たないユーザーには、メニューまたはメニュー項目は表示されない。

メッセージング・アプリケーション・プログラミング・インターフェイス(MAPI: Messaging Application Programming Interface):メッセージング・システムのコンポーネントとその動作を定義するアーキテクチャ。メッセージング・システムとコンポーネントの間のインターフェイスも定義する。

ミドルウェア(middleware):クライアントとサーバー間の相互作用をサポートするために必要なすべての分散ソフトウェアの総称。クライアント/サーバー・システムの中間に位置するソフトウェア、つまり、クライアントがサーバーからサービスを取得するための「接着剤」とみなされる。

モーダル(modal):特定の操作条件によって発生する限定的または制限付きの相互作用。通常、モーダルでは、ユーザーによる他のウィンドウの操作を制限するセカンダリ・ウィンドウが記述される。セカンダリ・ウィンドウは、プライマリ・ウィンドウまたはシステム全体に対してモーダルにすることができる。ユーザーは、あるモーダル・ダイアログ・ボックスを閉じなければ、そのアプリケーションの操作を続けることができない。

モード(mode):ERP のフォームでは、モードは次の 2 つの意味を持つ。

> テーブルおよびビジネス・ビューとフォームとの対話方法を管理するオペレーション修飾子。ERP のフォームのモードは、追加、コピー、更新の 3 つである。
>
> 異なる環境に対するフォーム生成の組織化を支援する任意設定。たとえば、Windows 環境に対して生成されたフォームをモード 1 に設定し、Web 環境に対して生成されたフォームをモード 2 に設定する。

モードレス(modeless):非限定的または無制限の相互作用。通常、モードレスでは、ユーザーによる他のウィンドウの操作を制限しないセカンダリ・ウィンドウが記述される。モードレス・ダイアログ・ボックスは画面に表示され、いつでも使用できるのみでなく、他のユーザー・アクティビティも許される。

複数階層アーキテクチャ(multitier architecture):さまざまなレベルの処理が許されるクライアント/サーバー・アーキテクチャ。階層では、なんらかの定義済みタスクを完了するために使用可能なコンピュータの台数が定義される。

イベント・ルール・ビジネス関数(named event rule):C プログラミングではなく、イベント・ルールを使用して作成され、カプセル化された再利用可能なビジネス・ロジック。「埋込みイベント・ルール」の反対語。「イベント・ルール」も参照。

NER:「イベント・ルール・ビジネス関数」を参照。

ネットワーク・コンピュータ(network computer):「パーソナル・コンピュータ」の反対語。ネットワーク・コンピュータの方が(理論上は)購入コストと所有コストが低く、単純である。基本的には、小規模な PC(メモリやディスクの容量がきわめて小さい)であり、ネットワーク・ブラウザ経由でネットワーク・ベースのアプリケーション(Java アプレット、ActiveX コントロール)へのアクセスに使用することができる。

ネットワーク・コンピューティング(network computing):通常は、クライアント/サーバーの次のコンピューティング・フェーズを指す。厳密な定義は明確になっていないが、通常はコンピューティング・リソースへの透過的アクセス、ブラウザ形式のフロントエンド、プラットフォーム独立型などの概念が含まれる。

自動採番(next numbers):新規 G/L 勘定科目、伝票、および住所などの項目の自動採番を制御するための機能。採番システムを指定し、トランスポジション・エラーと入力エラーを減らすために番号の増分方式を設定できる。

非オブジェクト・ライブラリアン・オブジェクト(non-object librarian object):オブジェクト・ライブラリアンによって管理されないオブジェクト。

数字(numeric character):データを表す 0～9 の値。「英数字」の反対語。

オブジェクト(object):データのみでなく、その操作に使用される構造体と関数を含む自己完結型エンティティ。ERP の場合のオブジェクトとは、そのツールセットによって作成されたソフトウェアのスペックに基づく再利用可能なエンティティである。「オブジェクト・ライブラリアン」も参照。

オブジェクト構成マネージャ(OCM:object configuration manager):ERP のオブジェクト・リクエスト・ブローカーと、ランタイム環境の制御センター。ビジネス関数、データ、およびバッチ・アプリケーションのランタイム・ロケーションを追跡する。これらのオブジェクトの 1 つが呼び出されると、オブジェクト構成マネージャ(OCM)は特定の環境とユーザーのデフォルト値および一時変更情報を使用して、そのオブジェクトにアクセスを転送する。

オブジェクト・エンベディング(object embedding):オブジェクトが他のドキュメントに埋め込まれている場合は、そのオブジェクトとそれが作成されたアプリケーションとの関連付けが維持される。ただし、そのオブジェクトの変更結果は、合成ドキュメントにしか保管されない。「オブジェクト・リンキング」も参照。

オブジェクト・ライブラリアン(object librarian):アプリケーションのビルドに再利用可能なすべてのバージョン、アプリケーション、およびビジネス関数のリポジトリ。これらのオブジェクトにアクセスするには、オブジェクト管理ワークベンチを使用する。

オブジェクト・ライブラリアン・オブジェクト(object librarian object):オブジェクト・ライブラリアンによって管理されるオブジェクト。

オブジェクト・リンキング(object linking):オブジェクトが別のドキュメントにリンクされている場合は、そのオブジェクトが格納されているファイルと、それが作成されたアプリケーションで参照が作成される。オブジェクトが合成ドキュメントから変更されるか、それが保存されているファイルを通じて直接変更されると、変更結果はそのアプリケーションのみでなく、リンク先にも反映される。「オブジェクト・エンベディング」も参照。

オブジェクト・リンキング・アンド・エンベディング(OLE:object linking and embedding):グラフィック、チャート、スプレッドシート、テキスト、またはサウンド・プログラムからのオーディオ・クリップなど、多様なアプリケーションからのオブジェクトを統合するための手段。「オブジェクト・エンベディング」、「オブジェクト・リンキング」も参照。

オブジェクト管理ワークベンチ(OMW:object management workbench):開発者にチェックアウトおよびチェックイン機能を提供するアプリケーションであり、ERP オブジェクトの作成、修正、および使用をサポートする。OMW は、複数環境(本稼働用と開発など)をサポートする。

オブジェクト・ベース・テクノロジー(OBT:object-based technology):オブジェクト指向テクノロジーの主要な原則であるクラス、多相性、継承、カプセル化のうち一部をサポートするテクノロジー。

オブジェクト指向テクノロジー(OOT:object-oriented technology):ソフトウェア開発後の手続き型プログラミングを、アプリケーション開発が簡素化された再利用可能なプログラミングに導く。オブジェクト指向は、クラス、多相性、継承、およびカプセル化という原則に基づくものである。

OCM:「オブジェクト構成マネージャ」を参照。

ODBC：「オープン・データベース・コネクティビティ」を参照。

OLE：「オブジェクト・リンキング・アンド・エンベディング」を参照。

OMW：オブジェクト管理ワークベンチ。

オンライン(online)：システムが持続的なコントロールを持つコンピュータ機能。J.D. Edwards システムが提供するフォームでの処理中は、システムとオンラインになっている。

オープン・データベース・コネクティビティ（ODBC：open database connectivity）：アプリケーションと各種データ・ソースとの間でデータを処理できるように、さまざまなテクノロジー用の標準インターフェイスが定義される。ODBC インターフェイスは、関数呼出しのセット、接続性の方法、およびデータ・ソースへのアクセスを定義するデータ・タイプの表示で構成される。

開放型システム間相互接続（OSI：open systems interconnection）：OSI モデルは、1980 年代初頭に国際標準化機構(ISO)により開発された。コンピュータとネットワーク設備間の相互接続のプロトコルと標準を定義している。

オペランド(operand)：「ブール論理オペランド」を参照。

出力(output)：コンピュータにより内部ストレージからプリンタやコンピュータ・フォームなどの外部デバイスに転送される情報。

出力待ち行列(output queue)：「印刷待ち行列」を参照。

パッケージ(package)：ERP オブジェクトは、デプロイメント・サーバーからのパッケージとしてワークステーションにインストールされる。パッケージは、そのワークステーションに必要なオブジェクトと、インストール・プログラムで検索されるデプロイメント・サーバー上の位置を示すという点で、部品表やキットとみなすことができる。デプロイメント・サーバー上のセントラル・オブジェクトの特定時点の「スナップショット」でもある。

パッケージ・ロケーション(package location)：パッケージとそのレプリケート・オブジェクト・セットが格納されるディレクトリ構造上の位置。通常は、¥¥deployment server¥release¥path_code¥package¥package name に置かれる。このパスの下のサブディレクトリに、パッケージ用のレプリケート・オブジェクトが格納される。パッケージがビルドまたは格納される場所を指すこともある。

パラメータ(parameter)：コマンドやプログラムに関連して指定する数値、コード、または文字列。パラメータは、コンピュータで追加入力として、またはコマンドやプログラムのアクションを制御する目的で使用される。

親/子フォーム(parent/child form)：あるフォーム上のアプリケーションの親/子関係を示すフォームのタイプ。フォームの左側には、親/子関係のビジュアル表現を表示するツリー表示がある。右側には、グリッドがブラウズ・モードで表示され、ツリー内の子項目のレコードが表示される。親/子フォームはドラッグ&ドロップ機能をサポートしている。

パーティショニング(partitioning)：アクセスするユーザーに近づけるために、データをローカル・サイトやリモート・サイトに分散するテクニック。データの各部分は、異なるデータベース管理システムにコピーできる。

パス・コード(path code)：特定のオブジェクト・セットを指すポインタ。パス・コードは次の要素の検索に使用される。

- セントラル・オブジェクト
- 複製オブジェクト

プラットフォーム独立型(platform independence):開放型システムとコンフィギュラブル・ネットワーク・コンピューティングの利点。単一コード・ベースで構成されるアプリケーションは、各種のサーバー・プラットフォームと SQL データベースで構成される TCP/IP ネットワーク上で実行することができる。

多相性(polymorphism):タイプの異なるソフトウェア・オブジェクトに対して、1 つのニーモニックを使用して類似する操作を実行できるようにする、オブジェクト指向テクノロジーの原則。

移植性(portability):同じアプリケーションを異なるオペレーティング・システムおよびハードウェア・プラットフォームで実行できること。

ポータル(portal):Web に情報とリンクを提供する、構成可能な Web オブジェクト。ポータルをホーム・ページとして使用でき、通常は Web ブラウザと組み合わせて使用する。

プライマリ・キー(primary key):テーブルの各ローを一意に識別するカラムまたはその組合せ。

印刷待ち行列(print queue):プリンタなどの出力デバイスに書き出すために投入したレポートなどのテーブルのリスト。テーブルは、書き出されるときまでスプールされる。テーブルが書き出されると、そのテーブル ID がシステム上のリストから削除される。

処理オプション(processing option):J.D. Edwards のレポート・システムの機能。この機能により、プログラムの機能を指示するためのパラメータを指定することができる。たとえば、処理オプションを使用して、特定のフォーム表示のデフォルト値を指定したり、情報がレポートに印刷されるときのフォーマットを制御したり、フォーマットの情報の表示方法を変更したり、開始日付を入力することができる。

プログラム一時修正(PTF:program temporary fix):磁気テープまたはフロッピー・ディスクに格納された形で組織が受け取る、J.D. Edwards ソフトウェアに関する変更。

プロジェクト(project):開発時にオブジェクトの組織化に使用されるオブジェクト管理ワークベンチ・オブジェクト。

パブリッシュされたテーブル(published table):「マスター」テーブルとも呼ばれ、他のマシンにレプリケートされるセントラル・コピー。「パブリッシャ」マシン上に存在する。データ・レプリケーション・パブリッシャ・テーブル(F98DRPUB)は、企業内でパブリッシュされた全テーブルとそれらに関連するパブリッシャを識別する。

パブリッシャ(publisher):パブリッシュされたテーブルを受け持つサーバー。データ・レプリケーション・パブリッシャ・テーブル(F98DRPUB)は、企業内でパブリッシュされた全テーブルとそれらに関連するパブリッシャを識別する。

プル・レプリケーション(pull replication):ERP でデータを個々のワークステーションにレプリケートする方法の 1 つ。この種のマシンは、ERP のデータ・レプリケーション・ツールを使用して、プル・サブスクライバとしてセットアップされる。プル・サブスクライバに変更、更新、および削除が通知されるのは、その情報を要求した場合のみである。要求は、通常は起動時に、プル・サブスクライバからデータ・レプリケーション保留変更通知テーブル(F98DRPCN)を保管するサーバーに、メッセージの形式で送信される。

除去(purge):システムのテーブルからレコードやデータを削除する処理。

QBE:「例示照会プログラム」を参照。

例示照会プログラム(QBE:query by example):グリッドの最上部に位置し、グリッドに表示されるデータの検索に使用する機能。

冗長性(redundancy):複数データベースのデータの正確なコピーを保管すること。

再生成可能(regenerable):ERP ビジネス関数のソース・コードは、スペック(ビジネス関数名)から再生成することができる。新規プラットフォームに使用するため、または新機能を追加したために、アプリケーションをリコンパイルするたびに、再生成処理が発生する。

リレーションシップ(relationship):リレーションシップによってテーブルや機能がリンクされ、アプリケーションやレポートに使用するビジネス・ビューを結合しやすくなる。リレーションシップは、インデックスに基づいて作成される。

リリース/リリース更新(release/release update):リリースには重要な新機能が含まれ、リリース更新には累積された修正やパフォーマンス強化などが含まれるが、新機能は含まれない。

レプリケート・オブジェクト(replicated object):セントラル・オブジェクトのコピーまたは複製セットは、ERP が稼働する各クライアントとサーバーに存在している必要がある。パス・コードは、これらのオブジェクトが格納されているディレクトリを示す。

実行(run):コンピュータ・システムにルーチン、トランザクションのバッチ処理、コンピュータ・プログラム指示を実行させること。

SAR:「ソフトウェア・アクション・リクエスト」を参照。

スケーラビリティ(scalability):物理的な容量またはリソース要件の拡大に伴って、ソフトウェアをサポートするソフトウェア、アーキテクチャ、ネットワーク、またはハードウェアを強化できる能力。マイクロプロセッサを追加することにより、パフォーマンスのレベルを高める機能。

検索/選択(search/select):値を検索して呼出し元フィールドに値を戻すために使用するフォームのタイプ。

選択(selection):J.D. Edwards メニューに表示される選択項目は、メニューからアクセスできる各機能を表す。選択するには、関連する番号を[Selection]フィールドに入力して[Enter]キーを押す。

サーバー(server):ネットワーク・ユーザー(つまりクライアント)にサービスを提供し、ネットワーク管理者に管理機能を提供するうえで不可欠な機能が組み込まれている。これらの機能の一部は、ユーザー・プログラムおよびデータのストレージ、ファイル・システムの管理機能である。1 台のサーバーでは、サービスを要求したユーザー全員をサポートできないことがある。特定のタスクを処理する専用サーバーの例として、ビジネスユニット・サーバー、アーカイブ・サーバー、アプリケーション・サーバー、データベース・サーバーがある。

サーブレット(servlet):拡張不可能なスクリプト・ソリューションなど、サーバー側のプログラミング実行に現在関連している問題の対処に使用される Java ベースのソリューションを提供する。また、Java ベースのサーバーにプラグインされた特定のインターフェイスに準拠するオブジェクトである。サーブレットはサーバーに対するもので、アプレットはクライアントに対するものである。

ソフトウェア(software):コンピュータに対して、実行するタスクとその実行方法を伝えるオペレーティング・システムおよびアプリケーション・プログラム。

ソフトウェア・アクション・リクエスト(SAR:software action request):AS/400 データベース上で、J.D. Edwards ソフトウェアの修正要求に使用する入力。

特殊文字(special character):データを表すために使用する記号。たとえば、*、&″、#、/など。「英数字」および「数字」の反対語。

スペック(specification):ERP オブジェクトの詳細記述。各オブジェクトには独自のスペック、つまり、アプリケーションのビルドに使用される名称がある。

スペック(Specs):「スペック」を参照。

スプール(spool):システムによって生成された出力を、印刷および処理されるまで待機させるために保管する機能。

スプールド・テーブル(spooled table):印刷されるまで待機している出力データまたは処理されるまで待機している入力データを保持するファイル。

SQL:「構造化照会言語」を参照。

スタティック・テキスト(static text):制御変数またはフィールドの隣に表示される短い記述テキスト。スタティック・テキストは、変数やフィールドが使用可能になっている場合は黒色で表示され、使用不可になっている場合は灰色で表示される。

オフライン処理(store and forward):トランザクション処理方式の 1 つで、クライアント・アプリケーションが処理を実行し、後でサーバー・アプリケーションに接続することによって処理を完了できるようにする。通常は、クライアント上に存在するデータをサーバーにアップロードする必要がある。

構造化照会言語(SQL:structured query language):リレーショナル・データベース・アクセスの業界標準として使用される第 4 世代言語。SQL を使用すると、データベースの作成、データベース内のデータの取込み、追加、修正または削除が可能である。SQL には制御フローのロジックが組み込まれていないため、完全なプログラミング言語ではない。

サブファイル(subfile):「詳細」を参照。

投入(submit):「実行」を参照。

サブスクライバ(subscriber):パブリッシュされたテーブルのレプリケート・コピーを受け持つサーバー。この種のサーバーは、サブスクライバ・テーブルで識別される。

サブスクライバ・テーブル(subscriber table):データ・レプリケーション・サブスクライバ・テーブル(F98DRSUB)は、パブリッシュされるテーブルごとにすべてのサブスクライバ・マシンを識別し、データ・レプリケーション・パブリッシャ・テーブル(F98DRPUB)と共にパブリッシャ・サーバーに保管される。

サブシステム・ジョブ(subsystem job):ERP では、サブシステム・ジョブは継続的に実行されるバッチ処理であり、ERP アプリケーションから独立して非同期で実行される。

集計(summary):ほとんどの明細が削除され、累計または合計形式で表現されたデータや情報。多くの J.D. Edwards システムには、特定のテーブルに保管された情報を集計するフォームとレポートが用意されている。「明細」の反対語。

システム(system):「アプリケーション」を参照。

システム・コード。J.D. Edwards システムおよびカスタマー・システムを表す番号。たとえば、01 は住所録のシステムコード。システム・コード 55～59 は、カスタマーによるカスタマー開発用に予約されている。システム・コードを使用して ERP 内で分類する。たとえば、ユーザー定義コード(UDC)の設定時には、それを分類する最適なシステム・コードを含める必要がある。アプリケーション、テーブル、メニューなどのオブジェクトの命名時には、オブジェクト名の 2 番目と 3 番目の文字はそのオブジェクトのシステム・コードを示す。たとえば、G04 は買掛管理メイン・メニューであり、04 はそのシステム・コードである、。

システム関数(system function):アプリケーションとレポートで処理を進めることができるように、ERP に用意されているプログラム・モジュール。

テーブル(table):ローとカラムで構成される 2 次元のエンティティ。データベース内の物理データは、すべてテーブルに保管される。テーブルのローには、関連情報のレコードが含まれる。例として、従

業員の氏名、住所、電話番号、年齢、給与が含まれる従業員テーブルのレコードがある。氏名は、テンプレートテーブルのカラムの一例である。

テーブル設計ツール（TDA：table design aid）データベース・テーブルの作成、修正、コピー、および印刷に使用する ERP GUI ツール。

テーブル・イベント・ルール(table event rules)：テーブルに対するアクション発生時に、自動的に実行されるデータベース・トリガー（またはプログラム）をアタッチするためのイベント・ルール。テーブルに対するアクションをイベントとみなす。ERP のデータベース・トリガーを作成するときには、どのイベントによってトリガーをアクティブ化するかを最初に決定する必要がある。次に、〈Event Rules Design〉を使用して、トリガーを作成する。ERP では、イベント・ルールをアプリケーション・イベントにアタッチできるが、この機能はアプリケーション固有のものである。テーブル・イベント・ルールでは、テーブル・レベルでの埋め込みロジックを提供します。

TAM：テーブル・アクセス管理(Table Access Management)。

TBLE：「テーブル」を参照。

TC：テーブル変換(Table conversion)。

TCP/IP：伝送制御プロトコル/インターネット・プロトコル(Transmission Control Protocol/Internet Protocol)。当初の TCP プロトコルは、多数の異なる伝送方法を使用してネットワークを相互接続する手段として開発された。TCP は、メッセージおよびデータの信頼性の高い配信を求め、エンド・システム間の接続を確立する方法を提供する。

TCP/IP サービス・ポート(TCP/IP services port)：特定のサーバー・アプリケーションによって、その設計意図に含まれたサービスを提供するために使用されるポート。アプリケーション・プログラマが名前を指定して要求できるように、一目でわかるポート番号を使用する必要がある。

TDA：「テーブル設計ツール」を参照。

TER：「テーブル・イベント・ルール」を参照。

端末 ID(Terminal Identification)：ワークステーションの ID 番号。特定の端末の端末番号か、またはこの ID が有効なプロファイルとなる特定のユーザーの IBM ユーザーID。[Header]フィールド：フォーム上部の[Skip to Terminal/User ID]フィールドを照会フィールドとして使用し、端末の番号か、またはリストの先頭にプロファイルを表示させたいユーザーの特定の IBM ユーザーID を入力する。このフォームに最初にアクセスしたときは、システムにサインオンしているユーザーのユーザーID が自動的に入力される。[Detail]フィールド：フォーム下部の[Terminal/User ID]フィールドには、同じ行にプロファイルが表示されているユーザーのユーザーID が含まれる。コードは、このウィンドウへのアクセスに使用されたユーザーまたは端末を識別する。

第 3 世代言語（3GL：third generation language）：タスクを完了させるために詳細情報を必要とするプログラミング言語。3GL の例には、COBOL、C、Pascal、および FORTRAN がある。

トークン(token)：オブジェクトの所有者を調べ、所有者以外がオブジェクト管理ワークベンチ内でこのオブジェクトをチェックアウトするのを防ぐために使用する。オブジェクトは、チェックアウトされるまで独自のトークンを保持する。チェックアウトされると、そのトークンはオブジェクトが配置されたプロジェクトに渡される。

トリガー(trigger)：トリガーを使用すると、データ辞書のデータ項目にデフォルトの処理を関連付けることができる。データ項目がアプリケーションやレポートに使用されると、そのデータ項目に関連付けられたイベントによってトリガーが起動される。ERP には、電卓、カレンダー、および検索フォームの 3 つのビジュアル・アシスト・トリガーも用意されている。

UBE：ユニバーサル・バッチ・エンジン(Universal batch engine)。

UDC 編集コントロール(UDC Edit Control)：ユーザー定義コード(UDC)の編集コントロールは、UDC テーブルに定義された特定値のみを許容するフィールドに使用する。UDC 編集コントロールを、データベース項目または辞書項目に関連付ける。UDC 編集コントロール・フィールドの隣には、ビジュアル・アシストのフラッシュライトが自動的に表示される。ビジュアル・アシストのフラッシュライトをクリックすると、アタッチされている検索/選択フォームに、そのフィールドの有効値が表示される。UDC 編集コントロールを作成するには、次を実行する。

- データ辞書内でデータ項目を特定の UDC テーブルに関連付ける。
- その UDC テーブルの有効値を表示する検索および選択フォームを作成する。

ユニフォーム・リソース ID（URI：uniform resource identifier）：インターネット・オブジェクトを名前または場所によって参照する文字列。URL は URI の一種である。

ユニフォーム・リソース・ロケータ（URL：uniform resource locator）：インターネットまたはイントラネット上のドキュメントのアドレス（位置）を指定する。URL にはドキュメントのプロトコルとサーバー名が含まれる。ドキュメントへのパスも含まれることがある。次に URL の例を示す。
http://www.jdedwards.com（J.D. Edwards のインターネット・アドレス）。

URI：「ユニフォーム・リソース ID」を参照。

URL：「ユニフォーム・リソース・ロケータ」を参照。

ユーザー定義コード（タイプ）(user defined code (type))：システム用に定義した意味を持つコード・テーブルの ID。たとえば、住所録の[Search Type]コード・テーブルの場合は ST である。J.D. Edwards システムはこのようなテーブルをいくつか提供しており、ユーザー独自のテーブルの作成および定義が可能である。ユーザー定義コードの旧称は「記述タイトル」。

ユーザー定義コード（UDC：user defined code）：ユーザーがコード記述に関連して定義して有効な値を割り当てることができる、ソフトウェア内のコード。汎用コード・テーブルを指す場合がある。この種のコードの例には、計量単位コード、都道府県名、従業員タイプ・コードがある。

UTB：ユニバーサル・テーブル・ブラウザ(Universal Table Browser)。

有効コード(valid code)：フィールドに入力できるデータのコード、金額、またはタイプ。入力した情報は、有効コードのリストと照合して確認される。

ビジュアル・アシスト(visual assist)：ユーザーがコントロールに属するデータを判別できるように、コントロールから起動できるフォーム。

用語一時変更(vocabulary override)：フォームやレポートのフィールド、ロー、またはカラムのタイトル・テキストを一時変更するための機能。

wchar_t：ワイド文字の内部タイプ。国際市場向けの移植可能プログラムを記述するために使用する。

Web クライアント(Web client)：インターネット・ブラウザが組み込まれているワークステーション。Web クライアントは、Web サーバーと ERP データをやりとりする。

Web サーバー(Web server)：IServer サービス、SQL Server、Java メニューとアプリケーション、インターネット・ミドルウェアを含むワークステーション。Web サーバーは、Web クライアントからデータを受信し、その要求をエンタープライズ・サーバーに渡す。エンタープライズ・サーバーは情報を処理してから Web サーバーに送り返し、Web サーバーは Web クライアントに送り返す。

WF：「ワークフロー」を参照。

ウィンドウ(window):「フォーム」を参照

ワークフロー(workflow):ワークフロー管理の一体化に従って、ワークフローは次のように定義される。「ビジネス・プロセスの一部または全体の自動化。自動プロセスの間は、ドキュメント、情報、またはタスクは 1 人の加入者から別の加入者へ渡されて、一連の手順規則に準じて実行される。」

ワークグループ・サーバー(workgroup server):通常は、マスター・データベース・サーバーからレプリケートされたデータのサブセットが保管されるリモート・データベース・サーバー。このサーバーでは、アプリケーションやバッチ処理は実行されない。ERP を実行しても、実行しなくてもかまわない(データをレプリケートするため)。

ワークスペース(workspace):Portal のメイン・セクション。ユーザーは複数のワークスペースにアクセスできる、それぞれのワークスペースは、異なる構成と独自のコンポーネントを持つ。

WWW (World Wide Web):インターネットのうち、テキスト、グラフィック、オーディオ、およびビデオを伝送できる部分。クライアントは、ローカル・アプリケーションまたはリモート・アプリケーションを起動することができる。

z ファイル(z file):オフライン処理中(ネットワークが切断されている状態)のユーザーの場合、スタティック・データや、オーダー処理のために有効であることが必要な他の重要な情報の編集は、ERP オフライン処理アプリケーションによって実行される。初期編集が完了すると、ERP によりトランザクションがワークステーション上のワークテーブルに保管される。これらのワークテーブルを Z ファイルと呼ぶ。ネットワーク接続が確立されると、Z ファイルがエンタープライズ・サーバーにアップロードされ、マスター・ビジネス関数によりトランザクションが再度編集される。その後、マスター・ビジネス関数によりトランザクション・ファイル内のレコードが更新される。